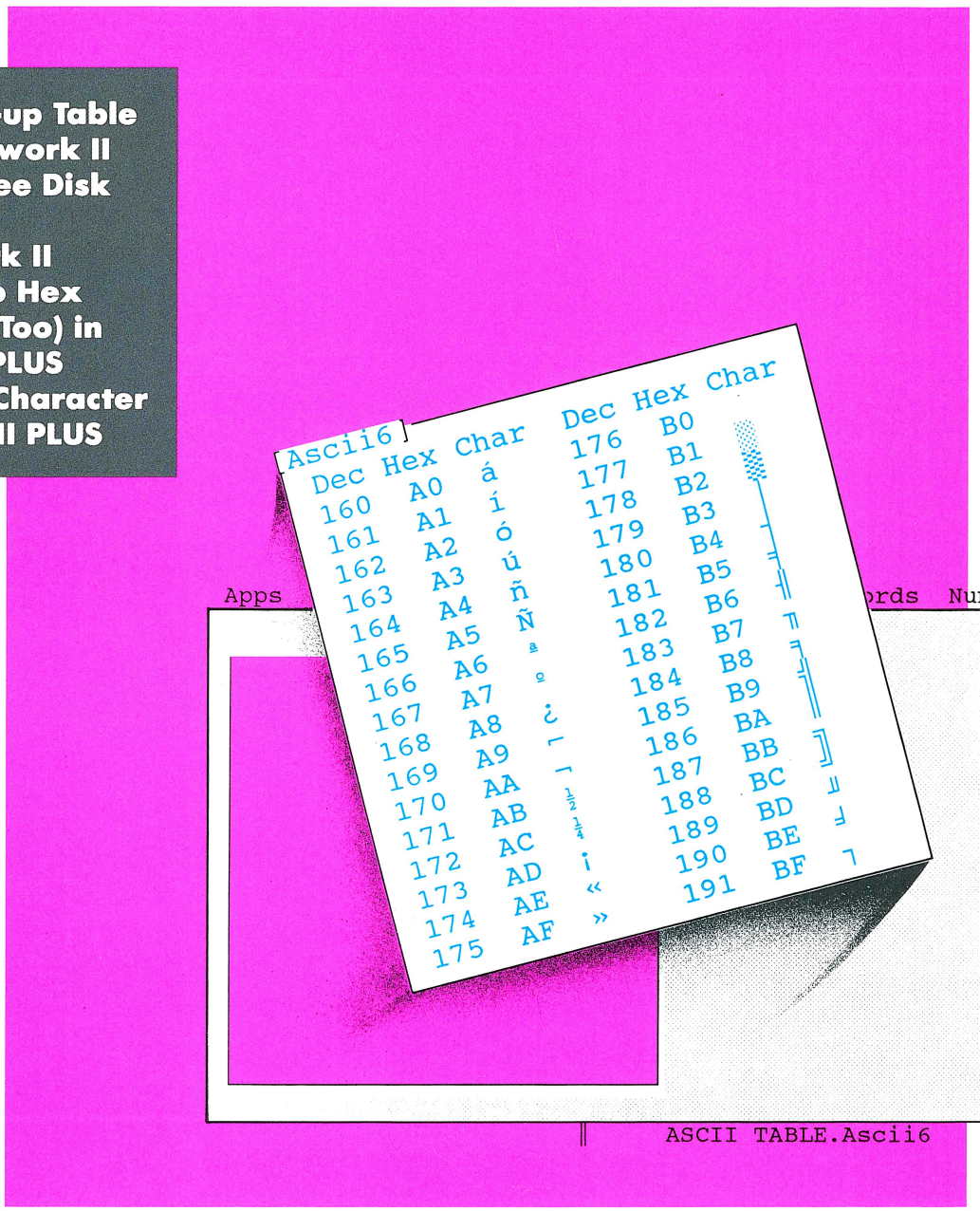
# TechNotes

ASHTON-TATE'S TECHNICAL JOURNAL
FEATURING dBASE AND FRAMEWORK

NOVEMBER 1987
$6.00

- **ASCII Pop-up Table for Framework II**
- **Finding Free Disk Space in Framework II**
- **Decimal to Hex (& Binary Too) in dBASE III PLUS**
- **Memo to Character in dBASE III PLUS**




ASHTON-TATE

# TechNotes

## TABLE OF CONTENTS

**NOVEMBER 1987**


| | | |
|---|---|---|
| **dBASE III PLUS** | Decimal to Hex (and Binary, Too)<br>*by Michael Carlisle and Kenneth N. Getz* | 5 |
| | Memo to Character Conversion<br>*by Rick Fillman* | 15 |
| | LAN Index Integrity<br>*by Phil Talsky and Chuck Litzell* | 25 |
| | Programmer's Casebook: Adding CR/LFs<br>*by Rodney Barbati, Kenneth N. Getz, and Chuck Litzell* | 29 |
| | dBASE III PLUS Tips | 36 |
| **Framework II** | ASCII Pop-Up Table<br>*by Kenneth N. Getz* | 40 |
| | Finding Free Disk Space<br>*by Kenneth N. Getz* | 55 |
| | Framework II Tips | 61 |
| **Departments** | Read Me First<br>*by Brenda Johnson-Grau* | 3 |
| | Anomalies | 66 |
| | On-Line Services | 70 |
| | Letters | 75 |

TechNotes

**Publisher**
Ashton-Tate
Support Products Center

*Jay Hanson*, Director
*Richard Goepel*, Manager

**Editor-in-Chief**
*Brenda Johnson-Grau*

**Editors**
*Kenneth N. Getz*
*James Bradbury*

**Associate Editor**
*Charles Litzell*

**Production Editor**
*Tom Pasley*

**Technical Staff**
*Larry Abare, Rozita Afar, Richard Augelli, Stanley Ballenger, Rodney Barbati, Richard Bunter, Mike Carlisle, Fred Crownover, Rick Fillman, James Chuang, Peter Gold, Mina Gussin, Scott Haynes, J. Mike Hedblom, Clint Helton, Errol Jackson, Matt Johnson, Gail Kulikowsky, Steve Kurasch, Chris LoVasco, Jeffrey McCrimon, Jerry Mendes, Vincent Mendillo, Jeffrey Moore, Keith Mund, Joseph Perret, John Regen, Cathy Shikatani, Joseph Stolz, Naoma Stow, Philip Talsky, Mitra Toossi, Hiram Vega, Ralph Wakefield, Mark Ware, Quinn Wildman, Roger Wegehoft*

**Contributing Editors**
*Paul Bukauskas (International), Liz Burke (MultiMate), Steve Silverwood (On-Line Services)*

**Production Staff**
*Gary Ashcavai, D.A. Gary Bare, Tim Dunne, Darrell Evers, Patty Pesek, Ron Powell, Ann O. Taono*

**Product Development**
*Forefront Development Center: Marilyn McMahon, Greg Stikeleather*
*Glendale Development Center*

# DISCLAIMER NOTICE

The information contained in this publication is designed to help you develop and improve your programming skills and to assist you in using Ashton-Tate products. Ashton-Tate assumes no responsibility whatsoever for the uses made of any of the software code in this issue, whether modified or unmodified.

If you discover what you believe is an anomaly (as defined under **Anomalies**) not presented in this issue or others, please send a detailed description of the possible anomaly to:

Ashton-Tate Software Support Center
20101 Hamilton Avenue
Torrance, CA 90502-1319

If the reported anomaly can be reproduced by our technicians, it may be added to a future issue of ***TechNotes.***

The policies that are described in ***TechNotes*** apply only to the United States (and possessions), Canada, and Mexico. Residents of other foreign countries should contact a local Ashton-Tate representative in their country for policy information and technical support.

Ashton-Tate and the authors will not be liable for special, incidental, consequential, indirect or other similar damages, even if we have been advised of the possibility of such damages. This means we are not responsible for damages or costs incurred as a result of loss of time, loss of data, loss of profits or revenues, or loss of use of the software. In addition, we are not responsible for damages or costs incurred in connection with obtaining substitute software, claims by others, inconvenience or similar costs.

No warranties, either expressed or implied, are made regarding the contents of this publication, its merchantability or fitness for any particular purpose, all of which are hereby expressly disclaimed. Note, however, that some states do not allow the exclusion or limitation of warranties or of direct, incidental or consequential damages, so these warranty limitations may not apply to you.

In no event will our liability for any damages to you or any other person ever exceed the price paid for your subscription to ***TechNotes***, regardless of any form of the claim.

Additional statements by agents, employees, distributors and dealers of Ashton-Tate do not constitute warranties by Ashton-Tate and do not bind Ashton-Tate.

. . . . . .

***TechNotes*** (ISSN 0883-6493) is an Ashton-Tate Support Products Center publication containing technical reference information on Ashton-Tate products.

Entire contents Copyright © Ashton-Tate Corporation 1987. All rights reserved.

Commercial reproduction of the material appearing in ***TechNotes*** is forbidden without prior written permission. Direct all requests for reproduction permission to Ashton-Tate Support Services, 20101 Hamilton Avenue, Torrance, CA 90502-1319, ATTN: Brenda Johnson-Grau.

***TechNotes*** is published monthly at a subscription price of $50.00 for one year. Add $15 for Canada and Mexico and add $25 for overseas. Back issues are available for $10.00 each (1985 issues) and $7.00 each (1986 and 1987 issues). ***TechNotes*** is published by Ashton-Tate, 20101 Hamilton Avenue, Torrance, CA 90502-1319.

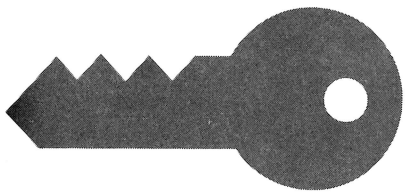
POSTMASTER: Send subscription inquiries and address changes to ***TechNotes*** Subscription Service, P.O. Box 3729, Escondido, CA 92025-0929.

**Subscription Inquiries**: Write ***TechNotes*** Customer Service, P.O. Box 3729, Escondido, CA 92025-0929. Call 800/828-2515 or 800/331-4164. From outside the U.S. call: 619/485-6535. When writing or calling, be sure that you have your current mailing label available. Your expiration date is listed on the far right of your mailing label.

dBASE, dBASE II, dBASE III, Ashton-Tate and Framework are registered trademarks and RapidFile, dBASE III PLUS, Framework II, RunTime, and FRED are trademarks of Ashton-Tate Corporation. Some of the trademarks of other companies mentioned in this issue appear for identification purposes only. MultiMate is a registered trademark and MultiMate Advantage, MultiMate GraphLink and the MultiMate Business Advantage Keyboard are trademarks of MultiMate International Corporation, an Ashton-Tate company. On-File is a trademark under license from Facts on File, Inc.

The Ashton-Tate Software Support Center's hours are 6:00 a.m. to 4:30 p.m., PST, Monday through Friday, except on Thursday when we close at 4:00 p.m. To reach our technicians for technical support, call (213) 329-0086. Be sure to have your software serial number ready for our operators.

# READ ME FIRST



By Brenda Johnson-Grau

Having trouble remembering the hexadecimal equivalent of 999,999,999,999,999? How about the binary? Giving you a headache? Instead of taking an aspirin, read "Decimal to Hex (and Binary Too)," by Michael Carlisle and Ken Getz. This article provides a set of dBASE III PLUS routines to convert binary to decimal, binary to hex, decimal to binary, decimal to hex, hex to binary, hex to decimal, hex to a hex reverse string (very helpful for use with dBASE III PLUS). Mike and Ken have also written a program that shows these routines in action.

For those moments when you suddenly have a desire to convert the data in your memo fields to character-type, just turn to "Memo to Character Conversion," by Rick Fillman. We've also provided some useful examples of how to use this program to count lines in a memo field or to print only active memo fields.

Maintaining healthy, happy indexes on a local area network isn't always easy. "LAN Index Integrity," by Phil Talsky and Chuck Litzell, shows you how to maintain good indexes and avoid bad ones. It turns out to be all in the locking....

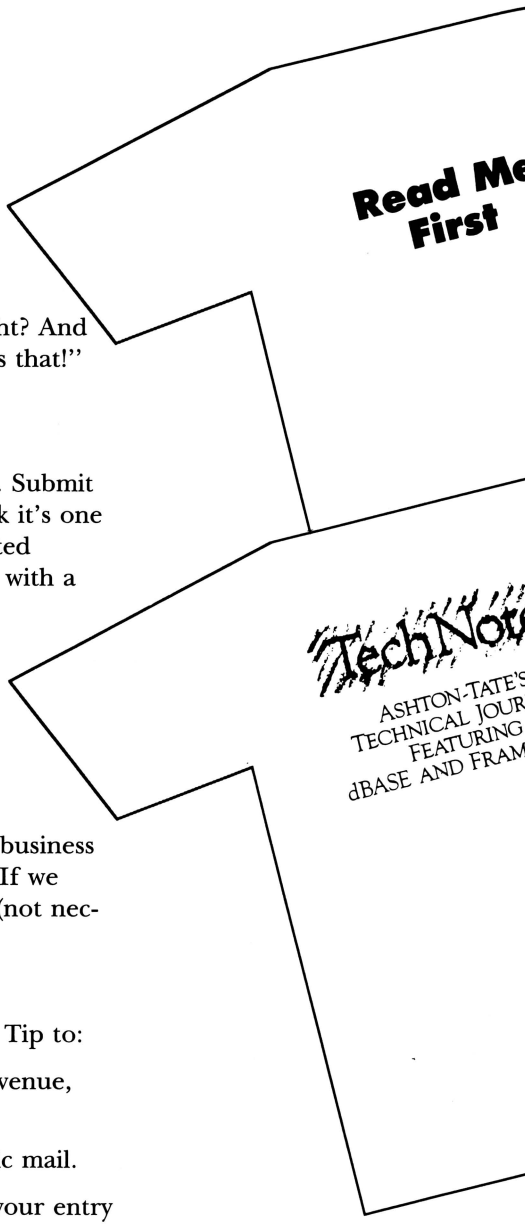
Our second Programmer's Casebook presents *three* solutions to the same problem. Importing can be traumatic if your SDF file-to-be-imported has no carriage return/linefeeds. Adding those necessary CR/LFs is easy with Makesdf.C by Chuck Litzell, Makesdf.PAS by Ken Getz, or Makesdf.BAS by Rodney Barbati. Just name your poison: C, Pascal, or BASIC.

In "ASCII Pop-up Table," Ken Getz has provided a macro that you can place in your Library Cabinet and use to create a handy pop-up table. Press {Alt-A} and the table appears on the desktop. You can then scroll through the pages and select either the ASCII values or their characters. It also demonstrates the use of keyfilters to control program flow. I know the amount of code for this program is rather daunting—especially for those of you who are not expert typists. Don't, though, forget that this and all the other programs in this issue are available from the Ashton-Tate forums on CompuServe and The Source. See On-Line Services for details.

Usually, you try to avoid DOS "Disk Full" errors. But, how do you figure out whether you have enough free disk space for an operation from within Framework II? With the program in "Finding Free Disk Space," by Ken Getz, you can easily find out how much space is free on a drive. This set of programs also showcases the *Framework II Developer's Toolkit,* which can help you to customize Framework II to your programming style.

*Coming soon in TechNotes:* More snaked columns, working around RETRY, fun with function keys, PROTECT, matching parentheses, hypertext, and the cumulative index for 1987.

# We've Got Your F4 (Size)!




You've been a loyal reader of ***TechNotes*** for at least a couple of issues, right? And every time we print a Usage Tip you think, "Heck, I've got one as good as that!" Well, now you can prove it.

**You (Almost) Can't Lose!**

Leave it to ***TechNotes*** to dream up a contest where almost everybody wins. Submit *your* favorite Usage Tip for dBASE III PLUS or Framework II. If we think it's one of the ten best that we receive that month, we'll send you the highly coveted ***TechNotes*** t-shirt in the color scheme of your choice—as long as it's white with a blue ***TechNotes*** logo.

How will we know that your Usage Tip is one of the ten best? Easy:

**It's going to be quick...**
**It's going to be handy...**
**It's going to be neat...**
**And most of all, it's going to be SHORT!**

That's right, leave those major applications programs for your consulting business and send us the short, juicy stuff. Our judges have short attention spans. If we think it's good, we'll send you a fashion statement that will last a lifetime (not necessarily yours, though).

**Here's how to enter:**

Fill out the form below (or a facsimile) and send it along with your Usage Tip to:

TechNotes Tips Contest, Attn: Ken Getz, Ashton-Tate, 20101 Hamilton Avenue, Torrance, CA 90502-1319.

Or, if you're an on-line user, you can send your entry to us over electronic mail.

On CompuServe, type GO EASY at any FUNCTION: prompt, and leave your entry via EasyPlex for Ken Getz, 76703,3033.

On The Source, choose SourceMail from the opening menu, and address your entry to SIG007.

---

I warrant that by submitting a tip or tips to the TechNotes Usage Tips Contest sponsored by Ashton-Tate Corporation, that I own all rights to any entry submitted or I have the full written consent of any owners to submit the entry, and I agree to hold Ashton-Tate harmless from any claims due to submission of entries. I understand that Ashton-Tate may publish or otherwise disseminate information about the winning entries.

Signature _____ Date _____

Name _____

Address _____

City _____ State _____ Zip _____

Daytime Phone Number _____ T-Shirt Size _____

| DECIMAL | BINARY | HEX | WORD-REVERSE HEX |
|---|---|---|---|
| 246809753 | OVERFLOW | OE B6 04 99 | 99 04 B6 OE |

# Decimal to Hex (and Binary, Too)

By Michael Carlisle and Kenneth N. Getz

Most compiled languages provide methods for dealing with numbers on the bit level, which makes converting to and from base 2 (binary), base 10 (decimal), and base 16 (hexadecimal) quite simple. With dBASE III PLUS, however, such conversions are more complicated. This article, and the seven procedures that go with it, will show you how to perform these conversions in dBASE. In addition, we've included a demonstration program that performs all the possible conversions for you on a full-screen display. (For a discussion of binary, decimal, and hexadecimal numbering systems, see almost any computer science textbook or the article "Bitwise Operations in dBASE III PLUS" in the October 1987 issue of ***TechNotes***.)

## Why You Need Conversions

If you've ever attempted to recover a damaged database header, you've needed to convert from hex to decimal. The main program presented in this article, Cnvtbase.PRG, was written with dBASE III PLUS file recovery in mind. When you examine your damaged database file with DEBUG, the bytes are represented in hexadecimal format. To complicate matters, hexadecimal numbers are stored in least-significant-byte-first format in dBASE III PLUS database files. As a result, the hexadecimal number

BC 12 3D EF

would be stored as

EF 3D 12 BC

in the database file. Each set of two hexadecimal digits forms a word, and words are stored backward in the file.

The article "Using DEBUG with dBASE" in the July 1987 issue of ***TechNotes*** describes several ways that DEBUG can help you understand the structure of your database files. Refer to that article for more information. Specifically, the quantities stored in the dBASE III PLUS file header that you might be interested in are these:

- The number of records in the file (bytes 4 through 7)
- The length of the file header in bytes (bytes 8 and 9)
- The length of the data records (bytes 10 and 11)

These are all stored in hexadecimal in byte-reverse order.

For example, let's assume that your database file contains 1,000 records, but when you LIST STRUCTURE or use RECCOUNT(), dBASE III PLUS shows only 850 records. You load DEBUG with Yourfile.DBF, and bytes 4 through 7 read

```
52 03 00 00
```

Since these bytes are in hex and backward, the actual hexadecimal number would be

```
00000352
```

This number is equivalent to 850 decimal. To repair your file you could use DEBUG to change bytes 4 through 7 to the actual number of records, 1000. But first you would need to calculate the hexadecimal equivalent of 1000, and then reverse the result, since these are the values that need to be inserted with DEBUG. This is where a program such as Cnvtbase.PRG can help.

## How the Conversions Work

The conversions fall into two categories. Some of them (Dec2bin, Dec2hex, Hex2dec) rely on successive divisions or multiplications to perform the changes. For example, to convert the hexadecimal number 23AB to decimal, you might do a calculation like that shown in Figure 1.

The Hex2dec routine does a similar series of operations.

Others (Hex2bin, Bin2hex) use a table-lookup to perform the conversion. Since the hexadecimal numbering system contains some digits that are not digits in the decimal numbering system, we place them in character strings where they can be accessed with a table-lookup. We use the two strings:

```
hexdigits = "0123456789ABCDEF"
```

and

```
bindigits = "0000 0001 0010 0011 " +;
            "0100 0101 0110 0111 " +;
            "1000 1001 1010 1011 " +;
            "1100 1101 1110 1111"
```

as our lookup tables. For example, to convert AC1F from hexadecimal to binary, we'd follow these steps:

```
For each character in the string
    Find its position in hexdigits ("A" gives
        us 11)
    Subtract 1 to find its value ("A" = 10)
    Calculate the offset in bindigits. (offset
        = 10 * 5 + 1)
    Extract the 4-character substring from
    bindigits. (SUBSTR(bindigits, offset, 4))
    Attach this string to the output string.
```

Although many languages provide internal functions or operators to perform these kinds of conversions, dBASE does not. dBASE does have a rich set of string operators, however, and the conversion methods used here take advantage of them.

## The Conversion Routines

Cnvtbase.PRG calls on seven conversion procedures:

- Bin2dec converts binary to decimal.
- Bin2hex converts binary to hexadecimal.
- Dec2bin converts decimal to binary.
- Dec2hex converts decimal to hexadecimal.
- Hex2bin converts hexadecimal to binary.
- Hex2dec converts hexadecimal to decimal.
- Hex2rev reverses a hexadecimal string.

Note that each of the first six procedures has an inverse procedure. For example, Dec2bin is the inverse of Bin2dec. The last routine, Hex2rev, is its own inverse; if you reverse a string twice, it's back to where it started.

Each procedure takes two parameters: an input and an output. The first parameter is the value to be converted. It may be a literal value or a memory variable. The second parameter must be a memory variable in which the result is to be stored.

```
11 * 1    +  ( "B" in hexadecimal = 11 in decimal)
10 * 16   +  ( "A" in hexadecimal = 10 in decimal)
 3 * 256  +
 2 * 65536
```

Figure 1

The data types of the parameters are determined by the type of conversion to be performed. All hexadecimal and binary values are character strings, and all decimal values are numeric. For example, Dec2hex requires a numeric value for its input parameter and a character string for its output parameter:

```
* ---Initialize hexival with enough space
* ---for longest possible hex value.
hexival = SPACE(16)
DO Dec2hex WITH 27, hexival
```

Dec2hex will store "1B" in the hexival memory variable. The inverse operation, converting 1B hex to decimal, is accomplished with a call to Hex2dec:

```
decival = 0
DO Hex2dec WITH hexival, decival
```

## Setup

To use Cnvtbase or the seven routines, you must first enter the source code into a text file named Cnvtbase.PRG. One of the first things Cnvtbase does is SET PROCEDURE TO Cnvtbase. An undocumented feature of dBASE III PLUS is that it allows a program to set itself as a procedure file. If you prefer, you could put each procedure in a separate file. If MODIFY COMMAND is your only text editor, you must do this, since a file containing all the procedures exceeds MODIFY COMMAND's limitation of 5,000 bytes.

## Using Cnvtbase

After you have created the file, type

```
DO Cnvtbase
```

at the dot prompt. Cnvtbase.PRG allows you to enter a value in decimal, hexadecimal, or binary and then displays the value in the other two bases. Hexadecimal numbers can be entered and displayed in byte-reversed order as well as the usual order. Figure 2 shows the screen after the decimal number 1234, has been entered. Note, the binary field is empty because the number is bigger than 256.

Cnvtbase stops on each line, where you may enter a value to be converted or press **Return** to go on to the next line. If you enter a value, it is converted and displayed, and then you can begin again at the top or press **Esc** to quit Cnvtbase.

The largest number that can be converted between decimal and hexadecimal by Cnvtbase.PRG is 999,999,999,999,999 or 038D73A4C67FFF hex, which should be large enough for most needs. Cnvtbase supports binary conversion for just one byte, or eight bits; so, if the value is greater than 255 decimal (FF hex), the binary equivalent will not be displayed.

To use the conversion procedures without Cnvtbase, first SET PROCEDURE TO Cnvtbase if you put all the procedures in one file. Then call the desired routine with the appropriate parameters as described above.

At the very lowest level, computers work in base 2, the binary numbering system. The hexadecimal numbering system is a more convenient way for humans to represent binary values since each hexadecimal digit represents four binary digits.

Many times it is necessary to know how to represent a value in more than one numbering system. For example, altering the record count in a dBASE III PLUS database file header, or converting a hexadecimal printer control code to decimal so that it can be used with the CHR() function.

Although it's possible to convert from one base to another with pencil and paper, why not let the computer do it for you? The programs in this article provide all the tools you need.

```
Decimal value      :                    1234

Hexadecimal value  : 04 D2

Byte-reversed value: D2 04

Binary value       :

Press ESCAPE to quit or any other key to repeat...
```

Figure 2

# Cnvtbase

```
* Program..:  Cnvtbase.PRG
* Author...:  Michael Carlisle and Kenneth N. Getz
* Date.....:  November 1, 1987
* Notes....:  Cnvtbase.PRG is an example program demonstrating the use of
*             hexadecimal, decimal, and binary conversion routines.
*
SET TALK OFF
SET PROCEDURE TO Cnvtbase
SET ESCAPE OFF
SET BELL OFF
CLEAR
numpict = "@Z 999999999999999"              && Picture for decimal.
hexpict = "@!R XX XX XX XX XX XX XX"        && Picture for hex.
binpict = "@R 9999 9999"                    && Picture for binary.
*
* ---Initialize the variables.
decnum = 0
hexstr = SPACE(20)
revstr = SPACE(20)
binstr = SPACE(9)
*
* ---Draw the original input screen.
@  9, 10 TO 17, 57 DOUBLE
@ 10, 12 SAY "Decimal value      :"
@ 10, 38 GET decnum;
         PICTURE numpict
@ 12, 12 SAY "Hexadecimal value  :"
@ 12, 33 GET hexstr
@ 14, 12 SAY "Byte-reversed value:"
@ 14, 33 GET revstr
@ 16, 12 SAY "Binary value       :"
@ 16, 44 GET binstr
CLEAR gets
DO WHILE .T.
   *
   * ---Reinitialize and redraw screen for next pass.
   decnum = 0
   hexstr = SPACE(20)
   revstr = SPACE(20)
   binstr = SPACE(9)
   @ 12, 33 GET hexstr
   @ 14, 33 GET revstr
   @ 16, 44 GET binstr
   CLEAR gets
   *
   * ---These four logical variables keep track of which blank was filled
   * ---by the user.
   gethex = .F.
   getnum = .F.
   getrev = .F.
   getbin = .F.
   @  7, 10 SAY "Press ESCAPE to quit."
   @ 19, 10
   *
   * ---The next block of code gets one of the possible formats for input.
   * ---In each case, if you enter nothing in a data field, the program
   * ---goes on to the next type of data.  The program will quit if you don't
```

```
* ---enter anything in any of the fields.  It also checks to see if you
* ---pressed Esc after each read, quitting if you did.
@ 10, 38 GET decnum;
         PICTURE numpict;
         RANGE 0, 999999999999999          && Only positive numbers.
READ                                       && Try for a decimal number.
IF READKEY() = 12
   EXIT
ENDIF
getnum = decnum <> 0
IF .NOT. getnum
   @ 12, 33 GET hexstr;
            PICTURE hexpict
   READ                                    && Try for a hex number.
   IF READKEY() = 12
      EXIT
   ENDIF
   gethex = LEN(TRIM(hexstr)) <> 0
   IF .NOT. gethex
      @ 14, 33 GET revstr;
               PICTURE hexpict
      READ                                 && Try for reverse-hex.
      IF READKEY() = 12
         EXIT
      ENDIF
      getrev = LEN(TRIM(revstr)) <> 0
      IF .NOT. getrev
         @ 16, 44 GET binstr;
                  PICTURE binpict
         READ                              && Try for a binary number.
         IF READKEY() = 12
            EXIT
         ENDIF
         getbin = LEN(TRIM(binstr)) <> 0
      ENDIF
   ENDIF
ENDIF
*
* ---If none were entered, quit.
IF .NOT. (gethex .OR. getnum .OR. getrev .OR. getbin)
   EXIT
ENDIF
@  7, 10
DO CASE
CASE getnum                                && We got a decimal number.
   revstr = ""
   hexstr = ""
   binstr = ""
   DO Dec2hex WITH decnum, hexstr
   DO Hex2rev WITH hexstr, revstr
   *
   * ---For the sake of this program, we've limited binary conversions
   * ---to 8 bits, though this was an arbitrary choice.  Anything up
   * ---to 254 bits should work.
   IF decnum < 256
      DO Dec2bin WITH decnum, binstr
   ENDIF
   @ 12, 33 GET hexstr;
            PICTURE hexpict
   @ 14, 33 GET revstr;
```

```
                PICTURE hexpict
         @ 16, 44 GET binstr;
                PICTURE binpict
         CLEAR gets
      CASE getbin                                  && We got a binary number.
         IF LEN(TRIM(binstr)) = 8                  && Skip it if not all 8
            revstr = ""                            &&    digits filled in.
            hexstr = ""
            revstr = ""
            DO Bin2dec WITH binstr, decnum
            DO Bin2hex WITH binstr, hexstr
            DO Hex2rev WITH hexstr, revstr
            @ 10, 38 GET decnum;
                     PICTURE numpict
            @ 12, 33 GET hexstr;
                     PICTURE hexpict
            @ 14, 33 GET revstr;
                     PICTURE hexpict
            CLEAR gets
         ENDIF
      CASE gethex .OR. getrev                      && We got some sort of hex.
         decnum = 0
         binstr = SPACE(9)
         IF gethex
            revstr = ""
            DO Hex2rev WITH hexstr, revstr
            @ 14, 33 GET revstr;
                     PICTURE hexpict
         ELSE
            hexstr = ""
            DO Hex2rev WITH revstr, hexstr
            @ 12, 33 GET hexstr;
                     PICTURE hexpict
         ENDIF
         DO Hex2dec WITH hexstr, decnum
         *
         * ---We're doing only 8-bit conversions.
         IF decnum < 256
            DO Hex2bin WITH hexstr, binstr
         ENDIF
         @ 10, 38 GET decnum;
                  PICTURE numpict
         @ 16, 44 GET binstr;
                  PICTURE binpict
         CLEAR gets
      ENDCASE
      @ 19, 10 SAY "Press ESCAPE to quit or any other key to repeat..."
      READ
      IF READKEY() = 12
         EXIT
      ENDIF
   ENDDO
   @  7, 10
   @ 19, 10
   CLOSE PROCEDURE
   SET CONFIRM OFF
   RETURN
   * ---EOP Cnvtbase.
```

```
PROCEDURE Bin2dec
PARAMETERS binstr, decnum
*
* ---This procedure converts numbers from binary to decimal by successive
* ---multiplications by 2.
*
multiplier = 1
counter    = LEN(TRIM(binstr))
decnum     = 0
DO WHILE counter > 0
   digit = VAL(SUBSTR(binstr, counter, 1))
   IF digit > 1
      decnum = -1
      EXIT
   ENDIF
   decnum     = decnum + digit * multiplier
   counter    = counter - 1
   multiplier = multiplier * 2
ENDDO
RETURN
* ---EOP Bin2dec.
```

---

```
PROCEDURE Bin2hex
PARAMETERS binstr, hexstr
*
* ---Procedure to convert from binary to hex.  This routine is independent
* ---of the number of binary places, so it will work with any length string
* ---of binary digits.
*
* ---The string constants below are used in the conversion process as psuedo-
* ---lookup tables.
* ---Hex     0    1    2    3    4    5    6    7    8    9    A    B    C    D    E    F
bindigits = "0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111"
hexdigits = "0123456789ABCDEF"
*
hexstr = ""                                  && Initialize the string.
newstr = binstr                              && Make a working copy.
length = LEN(TRIM(newstr))
*
* ---Find out how many groups of four digits are necessary to make the
* ---binary string an even multiple of four digits. Then create a
* ---string with the necessary amount of zeros into which to STUFF the
* ---binary string, ending up with a string that is a multiple of four
* ---digits long, ready for conversion to hex by 4-bit groups.
*
groups = INT(length / 4) + IIF(MOD(length, 4) <> 0, 1, 0)
nulstr = REPLICATE("0", groups * 4)
newstr = STUFF(nulstr, groups * 4 + 1 - length, length, newstr)
i = 0
DO WHILE i < groups
   segment  = SUBSTR(newstr, i * 4 + 1, 4)   && Get the current group.
   position = AT(segment, bindigits)         && Look for its position.
   IF position = 0                           && If not found,
      hexstr = ""                            &&    leave hexstr blank
      EXIT                                   &&    and quit.
   ENDIF
   *
   * ---Convert position to the number it represents, and then convert
```

```
   * ---to hex by lookup in the hex string.
   position = (position - 1) / 5 + 1
   hexstr   = hexstr + SUBSTR(hexdigits, position, 1)
   i        = i + 1
ENDDO
RETURN
* ---EOP Bin2hex.
```

---

```
PROCEDURE Dec2bin
PARAMETERS decnum, binstr
*
* ---This procedure converts from decimal to binary, using successive
* ---division by 2. It pads out the string with leading zeros to end
* ---up with a string exactly the length contained in the variable
* ---binplaces. If you need a longer binary string, just change this
* ---variable.
*
binplaces = 8

nulstr  = REPLICATE("0", binplaces)          && Create an empty string.
worknum = decnum                             && Local copy of the number.
IF worknum = 0                               && If the number is 0,
   binstr = nulstr                                && just send back nulstr.
   RETURN
ENDIF
binstr = ""
DO WHILE worknum > 0
   digit   = STR(MOD(worknum, 2), 1, 0)      && See if it's even/odd.
   worknum = INT(worknum / 2)                && Divide by 2 for next pass.
   binstr  = digit + binstr                  && Stick digit on string.
ENDDO
*
* ---STUFF the new string into the null string to guarantee "binplaces"
* ---characters.
binstr = STUFF(nulstr, binplaces + 1 - LEN(binstr),;
               LEN(binstr), binstr)
RETURN
* ---EOP Dec2bin.
```

---

```
PROCEDURE Dec2hex
PARAMETERS decnum, hexstr
*
* ---This procedure converts from decimal to hexadecimal. It performs this
* ---operation by successive divisions by 16. If you send in a 0 for decnum
* ---the program sends back "00" for hexstr. The routine also pads the
* ---most significant digit with 0 to make sure the number is composed of
* ---full words (two bytes each).
*
hexdigits = "0123456789ABCDEF"

worknum = decnum
IF worknum = 0
   hexstr = "00"
   RETURN
ENDIF
hexstr = ""
DO WHILE worknum > 0
   digit   = MOD(worknum, 16)
```

```
   worknum = INT(worknum / 16)
   hexstr  = SUBSTR(hexdigits, digit + 1, 1) + hexstr
ENDDO
IF MOD(LEN(hexstr), 2) <> 0                  && If the length is odd,
   hexstr = "0" + hexstr                     &&    add a "0".
ENDIF
RETURN
* ---EOP Dec2hex.
```

---

```
PROCEDURE Hex2bin
PARAMETERS hexstr, binstr
*
* ---This procedure converts from hex to binary in a manner very similar to
* ---the procedure Bin2hex above. It uses the string constants below as
* ---pseudo-lookup tables. If you need to use longer binary strings, change
* ---the value of the variable binplaces.
*
* ---Hex      0    1    2    3    4    5    6    7    8    9    A    B    C    D    E    F
bindigits = "0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111"
hexdigits = "0123456789ABCDEF"
binplaces = 8

nulstr  = SPACE(binplaces)
binstr  = ""
counter = LEN(TRIM(hexstr))
DO WHILE counter > 0
   position = AT(SUBSTR(hexstr, counter, 1), hexdigits)
   IF position = 0
      binstr = nulstr
      EXIT
   ENDIF
   binstr  = SUBSTR(bindigits, (position - 1) * 5 + 1, 4) + binstr
   counter = counter - 1
ENDDO
RETURN
* ---EOP Hex2bin.
```

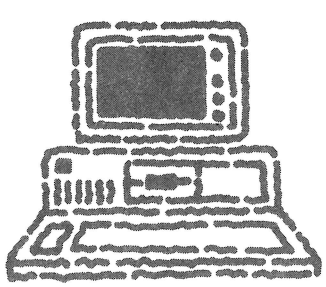
---

```
PROCEDURE Hex2dec
PARAMETERS hexstr, decnum
*
* ---This procedure converts a hexadecimal string to a decimal integer.
* ---It works by successive multiplications by 16. Starting at the right
* ---each character in the hex string is given a value based on its position
* ---in the string "hexdigits," and that positional value is multiplied by
* ---the correct power of 16 and added to a running total, the decimal
* ---value of the string. If any of the digits are illegal, decnum is given
* ---the value -1 (which would not normally be a legal value).
*
hexdigits = "0123456789ABCDEF"

multiplier = 1
counter    = LEN(TRIM(hexstr))
DO WHILE counter > 0
   position = AT(SUBSTR(hexstr, counter, 1), hexdigits)
   IF position = 0
      decnum = -1
```

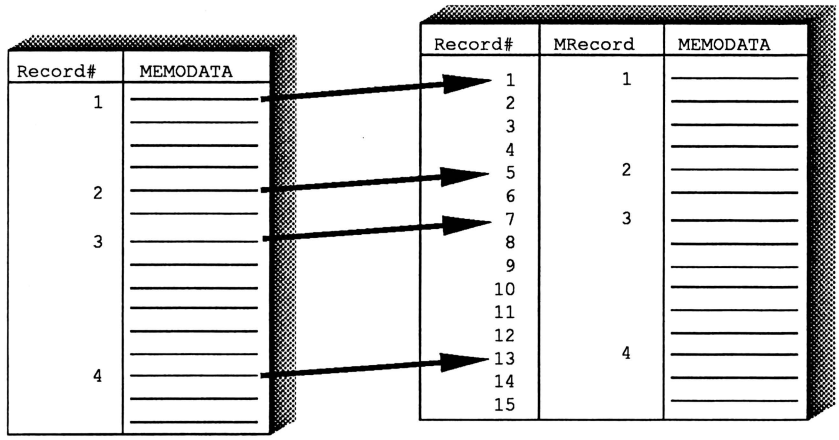
```
      EXIT
   ENDIF
   decnum     = decnum + (position - 1) * multiplier
   counter    = counter - 1
   multiplier = multiplier * 16
ENDDO
RETURN
* ---EOP Hex2dec.
```

---

```
PROCEDURE Hex2rev
PARAMETERS string1, string2
*
* ---Procedure to reverse the order of words in a hexadecimal string.
* ---This can be useful in deciphering the output of DEBUG. In physical
* ---terms, this routine takes groups of two characters and reverses their
* ---order in the string. For example, "ABCDEF" would become "EFCDAB".
*
* ---This routine depends on the input string having a length that is a
* ---multiple of two.
*
string1 = TRIM(string1)
i       = LEN(string1) - 1
DO WHILE i > 0
   string2 = string2 + SUBSTR(string1, i, 2)
   i = i - 2
ENDDO
RETURN
* ---EOP Hex2rev.
```



# Memo to Character Conversion




By Rick Fillman

An article in the May 1987 issue of ***TechNotes***, "Memo Fields Revisited," described the behavior of memo fields, including the limitations on manipulating memo field data under program control. Since it is not possible to use string operations with memo field data, you cannot search memo fields in a dBASE III PLUS program. Neither can you tell anything about the size of the memo data or even whether a memo field is empty.

As a result, if you want to manipulate memo field data, you must somehow convert it to character-type data. The program supplied here, Memo2dbf.PRG, does exactly that. Memo field data is APPENDed into a character field in a new .DBF file.

Although you can run Memo2dbf standalone, three example programs, each of which calls Memo2dbf as a subroutine, are included here. Each program demonstrates a way to use Memo2dbf with memo files. Memo2dbf must be available on the disk for any of these sample applications to run.

## What Memo2dbf Can Do

Memo2dbf places memo field data, one line at a time, into a character field in a separate database file for later use. This is what makes it possible to search, sort, and perform other string-type functions on the information that was contained in the memo fields. The example programs show some practical ways that you can put this new capability to use:

- Memotype.PRG types out only nonempty memo fields to the printer or screen.
- Memoline.PRG calculates the length of all memo fields, given the width of the page.
- Memochar.PRG copies the first line of memo field data to a character field.

## How Memo2dbf Operates

Memo2dbf is called with four parameters:

- The name of the memo field (in quotation marks or in a memory variable)
- The width of the memo field (a number between 10 and 254)
- The name of the file to be created (in quotation marks, with no extension)
- A numeric memory variable (to hold an exit status code), which must be initialized before you run the program

SET ALTERNATE TO <filename>, followed by SET ALTERNATE ON, sets up redirection of screen output to a file. The memo field data is written to the screen using LIST. Word wrapping occurs automatically within the bounds of the width setting passed in the second parameter.

The program then creates the target database file, with the name specified in the third parameter. The new file contains two character fields. The first is called "Mrecord" and contains the original record number of the memo field. The second field is given the name of the original memo field. The length of this field is determined by the second parameter. When the text file is APPENDed SDF, the memo field text data from the original database file is placed into this character field, using as many records as necessary for the entire contents.

The last parameter is a numeric memory variable into which Memo2dbf places an exit status code. This permits a program to call Memo2dbf as a subprocedure and handle errors that might occur. The error code is set to one of four values:

0 - No error occurred.
1 - No database in USE in workarea 1.
2 - The specified field does not exist or is not a memo field.
3 - The text line width is not between 10 and 254.

The database file with the memo field to be converted must be in USE in workarea 1. If no database file is in USE, the program aborts. In addition, two potentially large files are created by this process; so, plenty of free disk space will be required. The program does not attempt to check for disk space. The amount of free space required will be approximately twice the size of the present .DBT file.

## Results

Once the (formerly) memo data is in a character field, string searches and string operations of all kinds are possible. The new database has one record for each line of memo data. Even empty memo fields generate one record. The record number from the original file appears in the Mrecord field opposite the first line of each memo text block of records. Also, if any memo fields contained extra carriage returns at the bottom, these are represented as blank lines in the text file and thus blank records in the .DBF file.

## Limitations

The method used by this program, namely writing out to a temporary text file and APPENDing back into a .DBF file, is somewhat slow. Note that lots of free disk space is required. This method relies only on commands and procedures available within dBASE III PLUS.

## Memochar.PRG

At some point, you might decide to do away with memo fields entirely and convert as much of the existing memo field data as possible to character-type data. Memochar.PRG accomplishes this. The fixed-length (254 characters maximum) and no-word-wrap limitations of character-type data must be considered. Memo field data that is beyond the bounds of the specified line width will be lost during the conversion.

Memochar uses a file created by Memo2dbf to create a new database file with the memo field replaced by a character field. Memochar is called with five parameters:

- The name of the memo field to be converted (in quotation marks)
- The width of the character field to be created (between 10 and 254)
- The name of the file created by Memo2dbf (in quotation marks, with no extension)
- A numeric memory variable (to hold an exit status code), which should be initialized before you run the program
- The name of the new database to be created (in quotation marks, with no extension)

For example, to convert the memo fields of a file called Sales.DBF, use these commands:

```
error = 0
DO Memochar WITH "Sales", 65, "Temp", error,
   "Newsales"
```

Memochar tests for the existence of the file named by the third parameter. If the file exists, it will be used to create the new database. If it does not exist, Memochar calls Memo2dbf to create the file.

Memochar returns one of five possible exit status codes:

0 - No errors occurred.
1 - No database open in workarea 1.
2 - Field specified does not exist or is not a memo field.
3 - Width of field to create is outside of 10 to 254 range.
5 - Concatenated field names exceed 254 characters.

SET RELATION TO, SET FIELDS TO, and COPY TO are used to create the new database. Again, plenty of free disk space is needed. The COPY TO command must list all the fields in the original database. Since a command must not exceed 254 characters, the combined width of the field names from the original database, including

delimiting commas, must not exceed approximately 220 characters. An exit status code has been established to indicate that this limitation has been exceeded.

The new file will have the same structure as your original database, minus the memo field, but with a new character-type data field. It will have the same name as your original memo field and will contain the former memo field data.

The length of this new character field is specified by Memochar's second parameter. Since only one line of text from each memo field can be placed in the character field, text beyond the first line will be discarded.

## Memotype.PRG

The contents of memo fields can be "dumped" by the command LIST <memofield>. If there are many empty memo fields, however, this results in many blank lines. At least one line will be output for each record, regardless of whether the memo field is empty. If the destination is a printer, this can result in much wasted paper. The Memotype.PRG utility will print only those memo fields that have something in them.

Memotype deletes all records in a file created by Memo2dbf that do not contain a record number. This leaves only the first line of each memo field. A SET RELATION statement links the SELECT workarea 2 file to the SELECT workarea 1 file. The relation controls which records from the SELECT workarea 1 file are printed.

The program checks for the existence of text in the first line of the memo field only. Consequently, any memo fields with blank first lines are treated as if they were empty.

Memotype.PRG is called with five parameters:

- The name of the memo field to list (in quotation marks)
- The width of the character field created by Memo2dbf.PRG (between 10 and 254)
- The name of the database file created by Memo2dbf.PRG (in quotation marks, with no extension)
- A numeric memory variable (to hold an exit status code), which should be initialized before you run the program
- A logical value: .T. to print to printer, .F. to print to screen

For example, to print the memo fields of a file called Sales.DBF to your printer use these commands:

```
error = 0
DO Memotype WITH "Sales", 65, "Temp", error,
    .T.
```

The program checks for the existence of the file specified by the third parameter. If it does not exist, Memotype calls Memo2dbf to create the file. Memotype returns one of four exit codes in the fourth parameter:

0 - No errors occurred.
1 - No database open in workarea 1.
2 - Field specified does not exist or is not a memo field.
3 - Width of field to create is outside of 10 to 254 range.

## Memoline.PRG

A frustrating problem encountered when developing custom reports based on memo fields is that you can't tell in advance how many lines of text are contained in a memo field. Memoline.PRG will REPLACE into a field in the original database a number representing how many lines of text data are found in the memo field for each record.

The program requires that the user MODIFY STRUCTURE to add a numeric field (width 5, no decimals) called Memoline to the original database. The program REPLACEs the memo line count into this field for each record. If the memo field is empty, a zero is entered.

Memoline is called with four parameters:

- The name of the memo field (in quotation marks)
- The width of the character field in the database created by Memo2dbf (between 10 and 254)
- The name of the file created by Memo2dbf (in quotation marks, with no extension)
- A numeric memory variable to hold the exit status code, which should be initialized before you run the program

For example, to count the number of lines of text in memo fields of a file called Sales.DBF use these commands:

```
error = 0
DO Memoline WITH "Sales", 65, "Temp", error
```

Memoline checks for the existence of the file specified by the third parameter. If it does not exist, Memo2dbf is called to create the file.

Record number values from the resulting database are used to determine the number of lines of text. It is important to remember that the number of lines will be directly affected by the line-length setting in the second parameter.

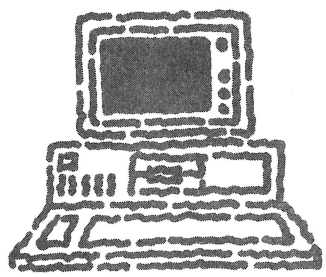
Memoline returns one of five exit status codes in its third parameter:

0 - No errors occurred.
1 - No database open in workarea 1.
2 - Field specified does not exist or is not a memo field.
3 - Width of field to create is outside of 10 to 254 range.
4 - The Memoline field does not exist in the database.

## Memo2dbf

```
* Program ....: Memo2dbf.PRG
* Author .....: Rick Fillman
* Date .......: November 1, 1987
* Version ....: dBASE III PLUS
* Notes(s) ...: Utility to bring MEMO field (.DBT file) data into a
*               dBASE database file (.DBF file) for analysis.
*               Make sure you have plenty of free disk space since
*                 1. Memotext.TXT temporary text file is created to
*                    hold contents of all Memo field data, and
*                 2. Memo2dbf.DBF will APPEND (import) the entire
*                    contents of Memotext.TXT
*               Thus, two possibly large temporary files will be created.
*               Floppy disk system users need to take special care.
*
*               Errors are returned in the parameter "error."
*
*                       0    no error
*                       1    no DBF in use
*                       2    wrong field type
*                       3    field width out of range
*
PARAMETERS memofield, memowidth, memofile, error
SET TALK OFF
SET SAFETY OFF
SET TITLE OFF
error = 0
* ---In case of setup error, ABORT program.
DO WHILE .T.
   *
   * ---Make sure a .DBF file is in USE in SELECT 1.
   SELECT 1
   IF LEN(DBF()) = 0
      error = 1
      EXIT
   ENDIF
   *
   * ---Make sure that field name represents a valid memo field.
   IF TYPE("&memofield") <> "M"
      error = 2
      EXIT
   ENDIF
    IF memowidth < 10 .OR. memowidth > 254
      error = 3
      EXIT
   ENDIF
```

```
   SET MEMOWIDTH TO memowidth
   *
   * ---Create output file from current database file.
    COPY TO Tempstru STRUCTURE EXTENDED
   SELECT 2
   *
   * ---Create .DBF structure with just two fields.
   USE Tempstru
    ZAP
    APPEND BLANK
    REPLACE Field_name WITH "mrecord",;
            Field_type WITH "C",;
            Field_len  WITH 8
    APPEND BLANK
    REPLACE Field_name WITH memofield,;
           Field_type WITH "C",;
           Field_len  WITH memowidth
    USE
    CREATE &memofile FROM Tempstru
   *
   * ---Clean up used temporary file.
   ERASE Tempstru.DBF
   *
   * ---List memo fields to a text file (and to the screen, too!)
   SET HEADING OFF
   SELECT 1
   SET ALTERNATE TO Memotext
   SET ALTERNATE ON
   LIST FIELDS &memofield
   CLOSE ALTERNATE
   *
   * ---Append from text file into memofile.
   SELECT 2
   USE &memofile
   ZAP
   APPEND FROM Memotext SDF
   *
   * ---Erase useless files.
   ERASE Memotext.TXT
   USE
   EXIT
ENDDO
SET TITLE ON
SET HEADING ON
SET MEMOWIDTH TO 65
RETURN
* EOP Memo2dbf.PRG
```



## Memochar

```
* Program ....: Memochar.PRG
* Author .....: Rick Fillman
* Date .......: November 1, 1987
* Version ....: dBASE III PLUS
* Notes(s) ...: Utility to move memo field data to a character-type field.
*               Source file is assumed to be in USE in SELECT 1 workarea.
*               A new dBASE file is created with the added field. (You
*               want to examine the results and then delete the original
*               file and rename the new file.) Dbfstruc.DBF is used internally
*               and then erased.
*
*               Errors are returned in the parameter "error."
*
*                       0    no error
*                       1    no DBF in use
*                       2    wrong field type
*                       3    field width out of range
*                       5    concatenated field names exceed 254 characters
*
PARAMETERS memofield, memowidth, memofile, error, newfile
SET SAFETY OFF
DO WHILE .T.
   * ---Make sure there's a datafile in USE.
   SELECT 1
   IF LEN(DBF()) = 0
      error = 1
      EXIT
   ENDIF
   IF TYPE("&memofield") <> "M"
      error = 2
      EXIT
   ENDIF
    IF memowidth < 10 .OR. memowidth > 254
      error = 3
      EXIT
   ENDIF
   SELECT 2
   IF .NOT. FILE("&memofile..DBF")
      DO Memo2DBF WITH memofield, memowidth, memofile, error
      IF error <> 0
         *
         * ---Pass back Memo2dbf's error number.
         EXIT
      ELSE
        SELECT 2
      ENDIF
   ENDIF
   USE &memofile
   * ---Construct a memory variable containing names of fields
   SELECT 1
   COPY TO Dbfstruc STRUCTURE EXTENDED
   SELECT 3
   USE Dbfstruc
   mfields = ""
   DO WHILE .NOT. EOF()
      *
      * ---Make sure this memory variable length doesn't exceed maximum
      IF LEN(mfields) >= 220
```

```
        error = 5
        EXIT
      ENDIF
      *
      * ---Build variable by repeated concatenation, not including Memo field.
      mfields = IIF(UPPER("&memofield") <> TRIM(field_name), ;
                  mfields  + "," + TRIM(field_name), mfields)
      SKIP
   ENDDO
   USE
   ERASE Dbfstruc.DBF
   *
   * ---Create index file to be used with SET RELATION.
   SELECT 2
   INDEX ON VAL(mrecord) TO &memofile
   SELECT 1
   *
   * ---Make fields from both files available.
   SET FIELDS TO B->&memofield &mfields
   SET RELATION TO RECNO() INTO B
   *
   * ---Create Newfile.DBF from fields in both database files
   COPY TO &newfile FIELDS &memofield &mfields
   SET RELATION TO
   SELECT 2
   USE
   EXIT
ENDDO
SELECT 1
SET INDEX TO
ERASE &memofile..NDX
SET SAFETY ON
RETURN
* EOP Memochar.PRG
```

## Memotype

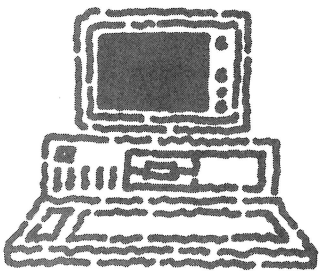
```
* Program ....: Memotype.PRG
* Author .....: Rick Fillman
* Date .......: November 1, 1987
* Version ....: dBASE III PLUS
* Notes(s) ...: Utility to eliminate empty Memo fields for printing.
*               Source file is assumed to be in USE in SELECT 1 workarea.
*               Only those that have content will print.
*
*               USAGE:
*                 DO Memotype WITH "<memofield>",;   && Name of memo field
*                                  memowidth,;       && Width of memo field
*                                  memofile,;        && Name of file created by
*                                                    &&    Memo2dbf.PRG
*                                  error,;           && Variable for errors.
*                                  logical_var       && Print to printer?
*
*         Errors are returned in the parameter "error."
*
*                       0    no error
*                       1    no DBF in use
*                       2    wrong field type
*                       3    field width out of range
*
```

```
*                Example:
*                  DO Memotype WITH "memo", 78, "Mymemos", error, .F.
*
PARAMETERS memofield, memowidth, memofile, error, printer
SET SAFETY OFF
SET TALK OFF
DO WHILE .T.
   *
   * ---Make sure a .DBF file is in USE in SELECT 1.
   SELECT 1
   IF LEN(DBF()) = 0
      error = 1
      EXIT
   ENDIF
   *
   * ---Make sure that field name represents a valid memo field.
   IF TYPE("&memofield") <> "M"
      error = 2
      EXIT
   ENDIF
    IF memowidth < 10 .OR. memowidth > 254
      error = 3
      EXIT
   ENDIF
   SELECT 2
   IF .NOT. FILE("&memofile..DBF")
      DO Memo2DBF WITH memofield, memowidth, memofile, error
      IF error <> 0
         *
         * ---Pass back Memo2dbf's error number.
         EXIT
      ELSE
        SELECT 2
      ENDIF
   ENDIF
   SET DELETED ON
   USE &memofile
   *
   * ---Mark for deletion all but the first record of any memo group
   DELETE FOR VAL(Mrecord) = 0
   SET RELATION TO VAL(mrecord) INTO A
   IF printer
      EJECT
      SET PRINT ON
   ENDIF
   LIST FIELDS mrecord, A->&memofield OFF FOR LEN(TRIM(&memofield)) <> 0
   SET PRINT OFF
   SET RELATION TO
   SET DELETED OFF
   RECALL ALL
   EXIT
ENDDO
USE
SELECT 1
SET SAFETY ON
SET TALK ON
RETURN
* EOP Memotype.PRG
```

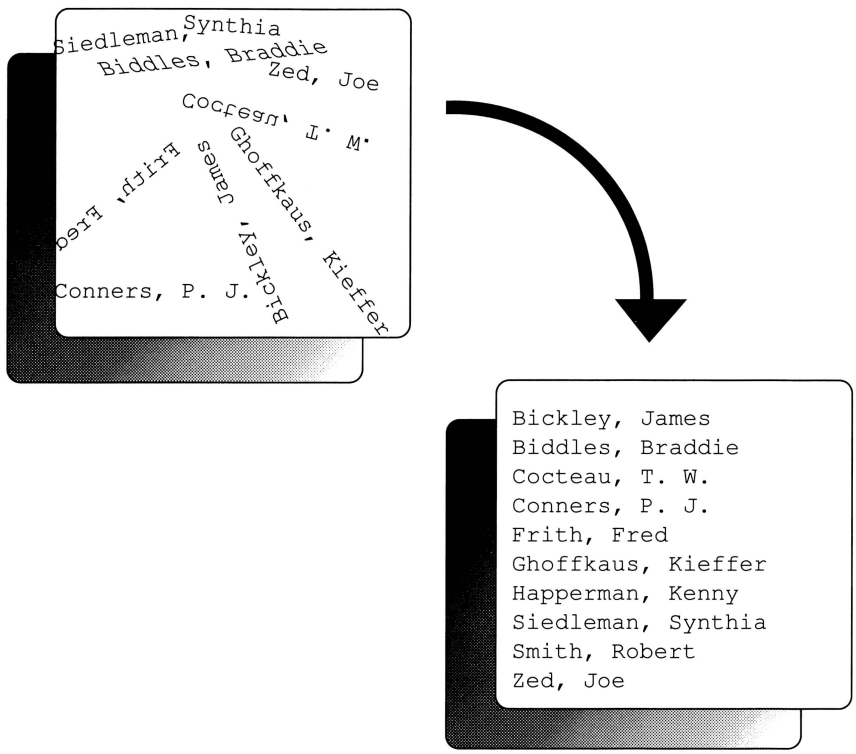
## Memoline

```
* Program ....: Memoline.PRG
* Author .....: Rick Fillman
* Date .......: November 1, 1987
* Version ....: dBASE III PLUS
* Notes(s) ...: Utility to return the number of lines of text in a memo
*               field. Source database file must contain a blank numeric field
*               named Memoline. The number of memo field text lines will
*               be entered into this field.
*
*               Source file must be in USE in SELECT 1 workarea.
*
*               USAGE:
*                 DO Memotype WITH "<memofield>",;    && Name of memo field
*                                  memowidth,;        && Width of memo field
*                                  memofile,;         && Name of file created by
*                                                     && Memo2dbf.PRG
*                                  error              && Variable for errors.
*
*            Errors are returned in the parameter "error."
*
*                      0    no error
*                      1    no DBF in use
*                      2    wrong field type
*                      3    field width out of range
*                      4    no numeric field Memoline
*
*
PARAMETERS memofield, memowidth, memofile, error
SET SAFETY OFF
SET TALK OFF
SET HEADING OFF
SET MEMOWIDTH TO memowidth
SELECT 1
DO WHILE .T.
   *
   * ---Make sure there's a datafile in USE.
   SELECT 1
   IF LEN(DBF()) = 0
      error = 1
      EXIT
   ENDIF
   IF TYPE("&memofield") <> "M"
      error = 2
      EXIT
   ENDIF
    IF memowidth < 10 .OR. memowidth > 254
      error = 3
      EXIT
   ENDIF
   *
   * --- Make sure Numeric field called Memoline has been created.
   IF TYPE("Memoline") <> "N"
      error = 4
      EXIT
   ENDIF
   SELECT 2
   *
```

```
   * ---Check that Memo2dbf.PRG has been previously run.
   IF .NOT. FILE("&memofile..DBF")
      DO Memo2DBF WITH memofield, memowidth, memofile, error
      IF error <> 0
         *
         * ---Pass back Memo2dbf's error number.
         EXIT
      ELSE
        SELECT 2
      ENDIF
   ENDIF
   USE &memofile
   *
   * ---Create index file to be used with SET RELATION.
   INDEX ON VAL(mrecord) TO &memofile
   SELECT 1
   SET RELATION TO RECNO() INTO B
   *
   * ---SKIP through source database
   GOTO TOP
   DO WHILE .NOT. EOF()
      ? recno()
      SELECT 2
      *
      * --- If not blank, note starting record number in Memo2dbf.DBF.
      mstart = IIF(LEN(TRIM(&memofield)) = 0, 0, RECNO())
      SELECT 1
      SKIP 1
      SELECT 2
      *
      * --- Note ending record number and calculate difference.
      mlength = IIF(mstart = 0, 0, RECNO() - mstart)
      SELECT 1
      SKIP -1
      *
      * --- Replace this value into Memoline field in source database file
      REPLACE Memoline WITH mlength
      SKIP
   ENDDO
   SELECT 2
   SET INDEX TO
   SET RELATION TO
   ERASE &memofile..NDX
   USE
   EXIT
ENDDO
SELECT 1
SET SAFETY ON
SET HEADING ON
RETURN
* EOP Memoline.PRG
```



# LAN Index Integrity




By Phil Talsky And Chuck Litzell

dBASE III PLUS programs which run on a local area network (LAN) require extra programming consideration if database files are to be accessed simultaneously by multiple users. The FLOCK() and RLOCK() functions, which control access to files and records, respectively, must be used to prevent data collisions.

The dBASE Administrator accesses the DOS 3.x file-handling services. When a disk file is to be updated, dBASE sends the data to DOS. Normally, DOS buffers data, writing to the disk file only when the file buffers are full, or the file is closed. Record-locking ensures that database records are updated immediately on the file server. It also prevents users from attempting to update the same record simultaneously.

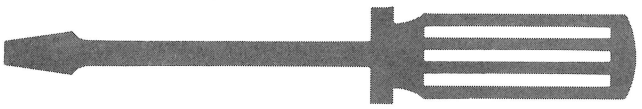
## Indexes at Risk

However, record-locking does not guarantee that index files open against a database will be updated properly. If a workstation updates key fields in a database, the index file must also be updated. Following normal operation, dBASE sends the data to be written in the index file to DOS. DOS will buffer the changes until forced to write them to disk. If several users are updating key fields in a database, there is a good chance that index files will be corrupted.

The problems with indexes arise because of the way that the dBASE Administrator handles them. On a LAN, dBASE maintains a picture of the index in

## Bad.PRG

```
* Program...:  Bad.PRG
* Author...:  Phil Talsky
* Date.....:  November 1, 1987
* Notes....:  Bad.PRG demonstrates the wrong way to use indexes in
*             dBASE III PLUS Administrator.
*
SET TALK OFF
ON ERROR DO Cretry
SET EXCLUSIVE OFF
SET SAFETY OFF
CLEAR
m1 = " "
counter = 0
*
* --- Testndx is indexed on F1 + F2.
USE Testfile INDEX Testndx
DO WHILE .T.
   counter = counter + 1
   m2 = STR(counter, 3)
   @ 10,  5 SAY "Initial (X to quit) ";
            GET m1;
            PICTURE "!"
   @ 11,  5 SAY "Number " + m2
   READ
   IF m1 = "X"
      EXIT
   ENDIF
   APPEND BLANK
   mrec = RECNO()
   SET INDEX TO Testndx
   GO mrec
   DO WHILE .NOT. LOCK()
   * DO nothing.
   ENDDO
   *
   * ---REPLACEs on fields F3 - F7 are arbitrary.
   REPLACE F1 WITH m1,;
           F2 WITH m2,;
           F3 WITH "XXXXX",;
           F4 WITH "YYYY",;
           F5 WITH "ZZZZ",;
           F6 WITH "OOOOOOOOOOO",;
           F7 WITH REPLICATE("Q-", 10)
   UNLOCK
ENDDO
* ---Display to show index corruption.
DISPLAY ALL F2, F1
ON ERROR
USE
RETURN
* ---EOP Bad.PRG.
```



## Good.PRG

```
* Program..:  Good.PRG
* Author...:  Phil Talsky
* Date.....:  November 1, 1987
* Notes....:  Good.PRG demonstrates correct methods to preserve index
*             integrity in dBASE III PLUS Adminstrator.
*
SET TALK OFF
SET EXCLUSIVE OFF
SET SAFETY OFF
CLEAR
m1 = " "
counter = 0
*
* --- Testndx is indexed on F1 + F2.
USE Testfile INDEX Testndx
DO WHILE .T.
   counter = counter + 1
   m2 = STR(counter, 3)
   @ 10,  5 SAY "Initial (X to quit) ";
            GET m1;
            PICTURE "!"
   @ 11,  5 SAY "Number " + m2
   READ
   IF m1 = "X"
      EXIT
   ENDIF
   *
   * ---Lock the file for APPEND BLANK.  APPEND BLANK issues an automatic
   * ---file lock, but then UNLOCKs immediately.  Therefore, you want to
   * ---force the file lock, to ensure that it remains locked.
   *
   * ---This method for file locking does not require an ON ERROR routine
   * ---to control the DO WHILE .NOT. FLOCK() section because other workstations
   * ---lock the file for a very short time.
   *
   DO WHILE .NOT. FLOCK()
   * ---DO nothing.
   ENDDO
   SET INDEX TO Testndx
   APPEND BLANK
   *
   * ---After adding the blank record, REPLACE only the key fields.
   * ---This ensures that the index is "locked" during its update.
   *
   REPLACE F1 WITH m1, F2 with m2
   *
   * ---After you replace the key fields, close the index.
   * ---This ensures that all changes are written to disk before the file
   * ---is unlocked.
   *
   CLOSE INDEX
   UNLOCK
   *
   * ---Immediately after UNLOCKing the file, LOCK the record to finish the
   * ---REPLACE statements.
   *
   DO WHILE .NOT. RLOCK()
```

```
   * ---Do nothing.
   ENDDO
   *
   * ---The following REPLACEs are arbitrary data.
   REPLACE F3 WITH "XXXXX",;
           F4 WITH "YYYY",;
           F5 WITH "ZZZZ",;
           F6 WITH "OOOOOOOOOOOO",;
           F7 WITH REPLICATE("Q-", 10)
   UNLOCK
ENDDO
* ---Display to show index integrity.
DISPLAY ALL F2, F1
USE
RETURN
* EOP Good.PRG
```

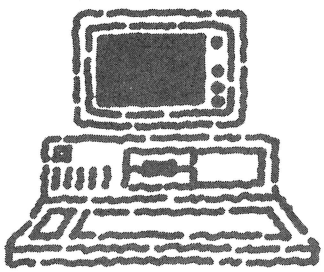
local workstation memory. When two workstations are updating the same index, if the local "picture" of the index is not refreshed after every update by any workstation, the changes made by the other workstations are overwritten. When this happens, users get records that drop out of the index or corrupted indexes.

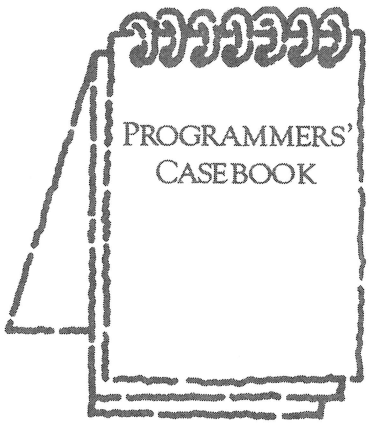
The program, Bad.PRG, when run by two users, consistently caused index corruption. (The file and field names have been changed to protect the innocent.)

To prevent index file corruption, the database should be locked to ensure that no other workstations have the index file open. FLOCK() doesn't lock an index file, but if you can lock the database file, you can be sure its indexes are not in use at another workstation. Index files should be closed immediately after key fields have been updated, in order to write the changes to the file server.

Bad.PRG was modified (producing Good.PRG) to guarantee that indexes could be updated without collision. The file is locked prior to the APPEND BLANK and not UNLOCKed again until the key fields have been REPLACEd and the index file has been closed to flush the buffers and write changes to the server. Then the record is locked with RLOCK() and the remaining fields are REPLACEd.

With these modifications, the program ran successfully with nine users entering data simultaneously.






# Adding CR/LFs

By Rodney Barbati, Kenneth N. Getz,
and Chuck Litzell

## The Problem

A colleague comes to you with a file that contains valuable contacts for your company. She has persuaded an acquaintance to make a copy of the file for her, and she wants you to read it into a dBASE III PLUS database and print some reports. She provides you with a list of the field names, types, and widths. The file is made up of fixed-length, ASCII-text records.

You create a dBASE III PLUS database file with the correct structure and use the APPEND FROM...SDF command to read the file. When you're through, though, your database has only one record in it. When you use the TYPE command to display the original file, you can see all the data. What's wrong?

The documentation for APPEND FROM...SDF says that in an SDF file each record is the same length and ends with a carriage return and a linefeed. If you list out an SDF file, it looks like the data was typed in columns.

The problem is that your colleague's file does not have carriage return/linefeeds at the end of each record. APPEND FROM...SDF doesn't know that, though. It treats the file as if it were one big record. It fills the first record but never begins the second.

## The Solution: Making an SDF File

What you need is a way to insert the carriage return/linefeed pairs into the file so that dBASE III PLUS can APPEND from the file. A program that did this would need to read the number of bytes in a record from the input file, write these bytes to an output file, and then write the carriage return and linefeed. It should cycle through these steps until the input file is exhausted.

Following are three versions of this program, which we called Makesdf. Chuck wrote the program in Microsoft C, version 4.0, and Ken wrote his in Turbo Pascal v3.01. Rodney's is written in interpreted BASIC.

All three programs follow the same general outline. Here is the program in pseudocode:

```
Check parameters. If anything is wrong, say so
and quit.
    There must be three parameters.
    Record length must be between 1 and 4000.
    Input file must exist.
    If output file exists, can it be
        overwritten?
Create output file.
While there are more bytes in the input file,
    Read up to a record's worth of bytes from
        the input file.
    Write the bytes read to the output file.
    Write a carriage return and linefeed to the
        output file.
    Display the number of records written so
        far.
Close both the input and output files.
```

The differences between compiled languages and BASIC are apparent in several ways. First, it is not possible to pass parameters to an interpreted BASIC program from the DOS command line. Therefore, the BASIC program asks the user to enter the filenames and record length.

Like dBASE III PLUS, BASIC is an interpreted language, and a significant part of the computer's resources are dedicated to translating program statements. To make the program run as fast as possible, the BASIC code is somewhat complicated, particularly where handling of file input and output is concerned. The more bytes that can be read

from the input file at each request, the faster the program will run. Care must be taken, however, not to read past the end of the file, since an incomplete record at the end of the file would be lost. Makesdf.BAS uses the LOF() function, which returns the number of bytes in the file, to determine how many complete records it can read. The complete records are read in chunks of up to 255 bytes to keep the program as fast as possible. If there is an incomplete record at the end of the file, it is read one byte at a time to avoid losing the last record.

## Setup for BASIC

If you are using the BASIC version, you can enter the BASIC program into a text file using an editor, or you may prefer typing it directly into the BASIC interpreter.

## Setup for Microsoft C

To compile Makesdf with Microsoft C, enter the C source code into a text file named Makesdf.C. The compiler should be installed and configured according to Microsoft's instructions. The following commands compile and link the program using Microsoft C, version 4.0.

```
MSC Makesdf;
LINK Makesdf;
```

If the source code was correctly entered, the result will be the file Makesdf.EXE. You can delete Makesdf.OBJ, an intermediate file created by the compiler.

## Setup for Turbo Pascal

To create the Turbo Pascal version, load Turbo Pascal, press E for Edit and enter Makesdf as the work file name. Enter the Pascal source code and press **Ctrl-KD** to save.

Since Makesdf takes command line options, it is easiest to compile to a .COM file, so press **O** for Options, **C** for .COM File, and **Q** to Quit. Next press **C** to Compile. When compilation is complete, Makesdf.COM will have been created. Press **Q** to quit Turbo Pascal.

## Usage

The C and Pascal versions of Makesdf work identically. They require three arguments that are passed on the command line. This way they can be RUN from within a dBASE III PLUS program without requiring input from the user. The calling syntax is:

```
Makesdf <infile> <outfile> <record length>
```

where <infile> is the name of the file having no CR/LFs, <outfile> is the name of the SDF file to create, and <record length> is the number of bytes per record. The <record length> must be in the range of 1 to 4000. If <outfile> already exists, Makesdf asks whether it is OK to overwrite it.

It isn't possible to pass arguments to an interpreted BASIC program through the DOS command line; so, that version of Makesdf prompts for the filenames and record length.

### Makesdf.C

```
/*  File.....: Makesdf.C                                                */
/*  Author...: Chuck Litzell                                            */
/*  Date.....: November 1, 1987                                         */
/*  Version..: Microsoft C, version 4.0                                 */
/*  Note(s)..: Program to insert CR/LFs after each record in an         */
/*             ASCII file containing fixed-length records and no CR/LFs.*/
/*             A file converted in this manner can be successfully      */
/*             APPENDed by dBASE III PLUS with the APPEND FROM...SDF    */
/*             command.                                                 */
/*                                                                      */
/*             Usage:  makesdf <infile> <outfile> <record-length>       */

#include <stdio.h>
#include <dos.h>

#define BELL 7
#define CR 0x0d
#define LF 0x0a
```

```
#define EOFC 0x1a
#define MAXSIZE 4000


char fbuf[4000];


main(argc,argv)
int argc;
char *argv[];
{
    FILE *ifd, *ofd;
    int reclen;
    register int i;
    char c;

/* Argument Checking                                                    */
/*   - Record count must be between 1 and 4000.                         */
/*   - input file must exist                                            */
/*   - if output file exists, make sure it's OK to overwrite            */
/*   - output file must be createable                                   */

    if (argc < 4)
       {
          fprintf(stderr,"Usage:  makesdf FLATfile SDFfile RecLen\n");
          exit(1);
       }

    if ((reclen = atoi(argv[3])) < 1 || reclen > MAXSIZE)
       {
          fprintf(stderr, "Record length must be between 1 and %d\n", MAXSIZE);
          exit(1);
       }

    if ((ifd = fopen(argv[1], "rb")) == NULL )
       {
          fprintf(stderr,"Can't open %s\n", argv[1]);
          exit(1);
       }

    if ((ofd = fopen(argv[2], "rb")) != NULL )
       {
          fprintf(stderr,"%s already exists. Overwrite [Y/N]? ", argv[2]);
          fclose(ofd);
          while ((c = toupper(getch())) != 'Y' &&  c != 'N')
             putch(BELL);
          putch(c);
          if (c == 'N')
             exit(1);
       }

    if ((ofd = fopen(argv[2], "wb")) == NULL )
       {
          fprintf(stderr, "Can't open %s for write.\n", argv[2]);
          exit(1);
       }


     /* Call Conversion Function */
     fprintf(stdout, "\nConverting %s to %s in %d-byte records.\n",
```

```
            argv[1], argv[2], reclen);
        convert(ifd, ofd, reclen);

        /* Cleanup and exit */
        fclose(ifd);
        fclose(ofd);
        exit(0);
}

/* convert() reads one record from the input file, writes it to the output   */
/* file and appends a CR/LF. This loop continues until the end of the file   */
/* is reached.                                                               */
/*                                                                           */
/* Partial records at the end of the file will be copied to the output, so   */
/* the user should always check the database file after APPENDing SDF,       */
/* particularly the last record in the database.                             */
/*                                                                           */
/* If the number of bytes written to the output differs from the number      */
/* read, an error has occurred, most likely disk full. convert() prints an   */
/* error message and quits.                                                  */

convert(in, out, len)
FILE *in, *out;
int len;
{
    int recc = 0;
    int nread;

    while ((nread  = fread(fbuf, 1, len, in)) > 0)
       {
       if (fwrite(fbuf, 1, nread, out) != nread)
          {
          fprintf(stderr, "\n%cError writing file. Check disk space.", 7);
          return(1);
          }
       else
          {
          ++recc;
          fputc(CR, out);
          fputc(LF, out);
          fprintf(stderr, "\rRecords: %d", recc);
          }
       }
     fputc(EOFC, out);
     return(0);
}
```

## Makesdf.PAS

```
PROGRAM Makesdf;

 (* Author...: Kenneth N. Getz                                              *)
 (* Date.....: November 1, 1987                                             *)
 (* Version..: Turbo Pascal version 3.x                                     *)
 (* Notes ...: Usage:                                                       *)
 (*                  makesdf infile outfile record-length                   *)
```

```
CONST
  MaxSize = 4000;

VAR
  Infile, Outfile : Text;
  OneRec : Array[1..MaxSize] OF Char;
  RecLen, RecCount : Integer;
  SavedX, SavedY : Integer;
  i : Integer;

FUNCTION ParamsOk (VAR RecLen : Integer) : Boolean;
  VAR ErrorCode : Integer;
      Ch : Char;

  BEGIN
    IF ParamCount <> 3 THEN
      BEGIN
        Writeln('Usage: makesdf FLATfile SDFfile RecLen');
        ParamsOk := False;
        Exit;
      END;
    RecLen := 0;
    Val(ParamStr(3), RecLen, ErrorCode);
    IF (ErrorCode <> 0) OR (RecLen < 1) OR (RecLen > MaxSize) THEN
      BEGIN
        Writeln('Record length must be between 1 and ', MaxSize:1);
        ParamsOk := False;
        Exit;
      END;
    Assign(Infile, ParamStr(1));
    Assign(Outfile, ParamStr(2));
    {$I-}
    Reset(Infile);
    IF IOResult <> 0 THEN
      BEGIN
        Writeln('Can''t open ', ParamStr(1));
        ParamsOk := False;
        Exit;
      END;
    Reset(Outfile);
    IF IOResult = 0 THEN
      BEGIN
        Write(ParamStr(2), ' already exists. Overwrite [Y/N]? ');
        REPEAT
          Read(Kbd, Ch);
          Ch := UpCase(Ch);
          IF NOT (Ch IN ['Y','N']) THEN
            Write(#7);
        UNTIL Ch IN ['Y','N'];
        Writeln(Ch);
      END;
    IF Ch = 'N' THEN
      BEGIN
        ParamsOk := False;
        Exit;
      END;
    Rewrite(Outfile);
    IF IOResult <> 0 THEN
      BEGIN
        Writeln('Can''t open ', Paramstr(2), ' for write.');
```

```
        ParamsOk := False;
      END;
    ParamsOk := True;
  END;

BEGIN
  IF ParamsOk(RecLen) THEN
    BEGIN
      Write('Converting ', ParamStr(1), ' to ', ParamStr(2), ' in ');
      Writeln(ParamStr(3), '-byte records.');
      RecCount := 0;
      Write('Records: ');
      SavedX := WhereX; SavedY := WhereY;
      WHILE NOT EOF(Infile) DO
        BEGIN
          { Read the records in. }
          FOR i := 1 TO RecLen DO
              Read(Infile,OneRec[i]);
          RecCount := RecCount + 1;
          GotoXY(SavedX, SavedY);
          Write(RecCount:4);
          { Write the records out. }
          FOR i := 1 TO RecLen DO
            Write(Outfile, OneRec[i]);
          Writeln(Outfile);
          IF IOResult <> 0 THEN
            BEGIN
              Writeln(#7);
              Writeln('Error writing file. Check disk space.');
              Halt;
            END;
        END;
      Close(Outfile);
      Close(Infile);
    END
END.
```

## Makesdf.BAS

```
10 ' Program ...: Makesdf.BAS
20 ' Author ....: Rodney Barbati
30 ' Date ......: November 1, 1987
40 ' Version ...: Interpreted BASIC
50 ' Note(s)....: Creates an SDF format file from an input file that
60 '              lacks a carriage return and linefeed after each record.
70 '
80 CLS
90 INPUT "Input  Filename"; INFILE$
100 INPUT "Output Filename"; OUTFILE$
110 INPUT "Record Length  "; RECLEN%
120 PRINT
130 ' ---Check record length.
140 IF RECLEN% < 1 OR RECLEN% > 4000 THEN PRINT "Record length must be between 1 and 4000":END

150 ' ---Check input file.
160 ON ERROR GOTO 190
170 OPEN INFILE$ FOR INPUT AS 1
180 GOTO 230
```

```
190 RESUME 200
200 PRINT "Can't open "; INFILE$
210 ON ERROR GOTO 0
220 END
230 ' ---Check output file.
240 ON ERROR GOTO 290
250 OPEN OUTFILE$ FOR INPUT AS 2
260 PRINT OUTFILE$; " already exists. Overwrite [Y/N]";
270 CHOICE$ = INPUT$(1)
280 IF CHOICE$ = "Y" OR CHOICE$ = "y" THEN GOTO 300 ELSE END
290 RESUME 300
300 CLOSE 2
310 OPEN OUTFILE$ FOR OUTPUT AS 2
320 ON ERROR GOTO 0
330 PRINT:PRINT
340 PRINT "Converting ";INFILE$;" to ";OUTFILE$;" in";STR$(RECLEN%);"-byte records."
350 NRECS% = INT(LOF(1) / RECLEN%)
360 FILELEN% = RECLEN%
370 WHILE RECLEN% > 255
380     REC%(L%) = 255
390     RECLEN% = RECLEN% - 255
400     L% = L% + 1
410 WEND
420 REC%(L%) = RECLEN%
430 RECNO% = 1
440 ON ERROR GOTO 720
450 LOCATE 10,1
460 PRINT "Record ";
470 WHILE NOT EOF(1) AND RECNO% <= NRECS%
480     LOCATE 10,8
490     PRINT STR$(RECNO%)
500     FOR T% = 0 TO L%
510         IF EOF(1) THEN 540
520             IN$ = INPUT$(REC%(T%), 1)
530             PRINT #2, IN$;
540     NEXT
550     PRINT #2,
560     RECNO% = RECNO% + 1
570 WEND
580 IF EOF(1) THEN GOTO 730
590 ON ERROR GOTO 720
600 WHILE NOT EOF(1)
610    N% = FILELEN%
620    WHILE NOT EOF(1) AND N% > 0
630       IN$ = INPUT$(1, 1)
640       PRINT #2, IN$;
650       N% = N% - 1
660    WEND
670    PRINT #2,
680    LOCATE 10,8
690    PRINT STR$(RECNO%)
700 WEND
710 GOTO 730
720 RESUME NEXT
730 ON ERROR GOTO 0
740 CLOSE
750 END
```

By J. Mike Hedblom

## Sending CHR(0) to the Printer

By Chuck Litzell

dBASE III PLUS is unable to send CHR(0) to the printer. Since dBASE uses CHR(0) internally to mark the end of strings, it processes right up to, but does not include, the CHR(0).

This is a problem only if you need to send control codes that include CHR(0) to the printer. Many printers ignore the high bit when processing escape codes. Therefore, sending CHR(128) in place of CHR(0) might work because CHR(128) is the equivalent of CHR(0) with bit seven turned on.

If CHR(128) does not produce the desired results with your printer, however, you can use the following DEBUG script to create a .BIN file that will send CHR(0) to your printer. Using a text editor or the dBASE MODIFY FILE command, create a text file, named Prnull.DBG, with the lines that follow. Note the empty line five lines from the bottom; it's important!

```
N    PRNULL.BIN
A    100
PUSH AX
PUSH DX
XOR  DX, DX
MOV  AX, 0500
INT  21
POP  DX
POP  AX
RETF

RCX
C
W
Q
```

To create Prnull.BIN, you must have access to DEBUG.COM, which you will find on your DOS Supplemental Program diskette. Change to the drive and directory in which you want to create Prnull.BIN and enter

```
DEBUG < Prnull.DBG
```

This command will create Prnull.BIN. If you don't get the DOS prompt rather quickly, there is an error in your DEBUG script. Press **Return** until you see a dash (DEBUG's prompt), and type **Q** and then press **Return** to quit DEBUG. When Prnull.BIN has been successfully created you can erase Prnull.DBG.

Using Prnull.BIN from within dBASE III PLUS is simple. First LOAD Prnull. Any time you need to print a CHR(0), just CALL Prnull. The following example sets the form length to seven inches on an Epson FX-80 printer.

```
* Set Form Length to 7 inches
*      on Epson FX-80 printer
* (ESC C NUL 7)
LOAD Prnull
SET PRINT ON
SET CONSOLE OFF
?? CHR(27) + C
CALL Prnull
?? CHR(7)
SET PRINT OFF
SET CONSOLE ON
RELEASE MODULE Prnull
```

## Overstriking with @...SAY

It is possible to produce overstriking on a printer with the @...SAY command in dBASE III PLUS. The documentation warns that, after executing SET DEVICE TO PRINT, an @...SAY command with row or column coordinates less than the coordinates of the previous @...SAY will cause a page eject. This is true of row coordinates. However, it is possible to @...SAY at a lesser column coordinate in the same row or even at the same coordinate. For example, the following commands produce underlined output.

```
SET DEVICE TO PRINT
@ 10,20 SAY "UNDERLINE"
@ 10,20 SAY "_________"
SET DEVICE TO SCREEN
```

## Justifying and Centering Strings

By Chuck Litzell

Frequently, there is a need to center or right justify a message on the screen or in a report. The usual method is to add to the left of the string the number of spaces required to offset the message to the desired position. The dBASE III PLUS code to accomplish this is short, though somewhat complicated owing to the mathematics involved.

Use this expression to center a string:

```
string = IIF(width > LEN(string),;
       SPACE((width - LEN(string)) / 2),;
       "") + string
```

To right justify a string, use

```
string = IIF(width > LEN(string),;
       SPACE(width - LEN(string)),;
       "") + string`
```

Often, reports and screen displays have headings that include a left-justified string, a centered string, and a right-justified string. Title.PRG (Figure 1) is a procedure that builds such a heading. The three strings are passed as parameters, along with the width of the desired heading. The resulting string is returned in the first parameter.

## Trapping the Esc Key

In dBASE III PLUS, version 1.1, SET ESCAPE OFF is ignored by the WAIT, INPUT, and ACCEPT commands. If the user presses **Esc** in response to any of these commands, dBASE III PLUS responds with the "Cancel, Ignore, or Suspend?" message, and the user can cancel the application program.

The work-around is to add an ON ESCAPE command to the application:

```
ON ESCAPE DO Esc_back
```

Esc_back.PRG needs just one line:

```
RETURN
```

With the ON ESCAPE active, pressing **Esc** at a WAIT, INPUT, or ACCEPT command causes the program to resume at the line following the WAIT, INPUT, or ACCEPT command with no error messages.

In version 1.0, pressing **Esc** in response to an INPUT statement with SET ESCAPE OFF hangs the system, while an **Esc** at a WAIT statement returns to the dot prompt.

## ASSIST

In ASSIST, it is not possible to specify the .NOT. (or FALSE) of a logical field in a search condition. When a logical field is specified as the search condition, ASSIST uses .T. as the search condition. This is true of versions 1.0 and 1.1 of dBASE III PLUS.

## BROWSE

The **Ctrl-Home** and **F10** keys behave slightly differently in BROWSE. Both keys call the menu bar to the screen, but only **Ctrl-Home** removes it. This is true of both versions of dBASE III PLUS.

## BROWSE

BROWSE is extremely slow paging through a database file if the file is indexed and the index contains many identical entries. The wait to move from one page to the next is better measured in minutes than seconds and may appear to have locked the machine. The amount of time necessary is proportional to the amount of records with identical keys in the index file. This occurs in both versions of dBASE III PLUS.

## CREATE REPORT

In CREATE REPORT, the **Columns:Decimal places** value defaults to the number of decimal places specified by SET DECIMALS TO when SET FIXED is ON; ignoring the number of decimal places specified in the file structure. The documentation for CREATE REPORT incorrectly states that the **Columns:Decimal places** value defaults to the number of decimal places specified in the file structure. This is true of both versions of dBASE III PLUS.

## "EVAL Work area" Messages

The dBASE III PLUS "EVAL work area" is used by any expression in any command that allows any type of expression or condition. Commands that contain a condition or expression invoke the expression evaluator and use the evaluation workarea. The error message "Internal error: EVAL work area overflow" occurs when a command is too complex to be evaluated within the workarea's allocated space. This is true of all versions of dBASE III PLUS.

```
* Program ...: Title.PRG
* Versions ..: dBASE III PLUS v1.0, 1.1
* Note(s) ...: Generates a report or screen title line with text
*              left justified, centered and right justified.
*              The result is returned in the first parameter, ltext.
*
*              The calling program must ensure that the width
*              specified is sufficient to contain the three
*              strings without overlap.  No error will result
*              if there is overlap, but the result string will be
*              erroneous.
*
*              Example:
*                 pagen  = 1
*                 header = DTOC(DATE())
*                 DO Title WITH header, "Employee Master Report",;
*                              "Page " + LTRIM(STR(pagen)), 80
*                 @ 1, 0 SAY header
*
PARAMETERS ltext, ctext, rtext, width
*
*          ltext  ::=  text to be left justified.
*          ctext  ::=  text to be centered.
*          rtext  ::=  text to be right justified.
*          width  ::=  width of output string.
*
  tmpstr = RIGHT(SPACE(width) + TRIM(rtext), width)
  tmpstr = STUFF(tmpstr, (width - LEN(TRIM(RTRIM(ctext)))) / 2,;
           LEN(TRIM(RTRIM(ctext))), TRIM(RTRIM(ctext)))
  ltext  = STUFF(tmpstr, 1, LEN(TRIM(RTRIM(ltext))), TRIM(RTRIM(ltext)))
  RETURN
* EOP Title.PRG
```

Figure 1

## F1 Key

The **F1** function key can be used to exit an active @...GET if the @...GET is not within a format file opened with SET FORMAT TO. Changes made prior to pressing **F1** are saved as if **Ctrl-W** were pressed.

However, even though **F1** exits and saves the data, it does not produce exit-key values that are consistent with other dBASE III PLUS write operations. Write operations always save the data and produce an exit-key value that READKEY() interprets as data changed, regardless of whether data have been altered. When **F1** is used to exit a full-screen operation, READKEY() returns either 36 for no change in data or 292 when data have changed. To be consistent, **F1** should not produce 36 since it means non-updated data.

Pressing **F1** in an active screen format moves the cursor to the beginning of the current field without exiting the screen. This occurs in all versions of dBASE III PLUS.

## LOOP

The command verb LOOP does not appear in the second edition documentation index. An explanation of LOOP can be found on page U5-114 under the DO WHILE command.

## REPLICATE()

Once the error message "Execution error on REPLICATE(): String too large." [88] is produced, it does not reoccur until a valid numeric expression is included in REPLICATE(). For example,

```
* ---The next line produces the error.
mvar = REPLICATE("A",32766)
* ---In the next line no error is produced.
mvar = REPLICATE("A",32767)
mvar = REPLICATE("A",10)
* ---The next line produces the error.
mvar = REPLICATE("A",32767)
```

The result of REPLICATE() is a null string whenever the second argument is too large, regardless of whether an error message is returned. This is true of versions 1.0 and 1.1 of dBASE III PLUS.

## REPORT FORM

**CREATE REPORT Options:Double space report** causes double spacing between records only. All other portions of the REPORT FORM, including wrapped fields, are single spaced. This is true of both versions of dBASE III PLUS.

## SET COLOR TO

It is not documented that the U and I arguments of SET COLOR TO, which are for monochrome monitors, set the color to black when used on a color monitor. For example, on a color monitor

```
SET COLOR TO U/I
```

is equivalent to

```
SET COLOR TO N/N
```

## SET ESCAPE OFF

**Esc** terminates an active READ regardless of the status of SET ESCAPE in both versions of dBASE III PLUS.

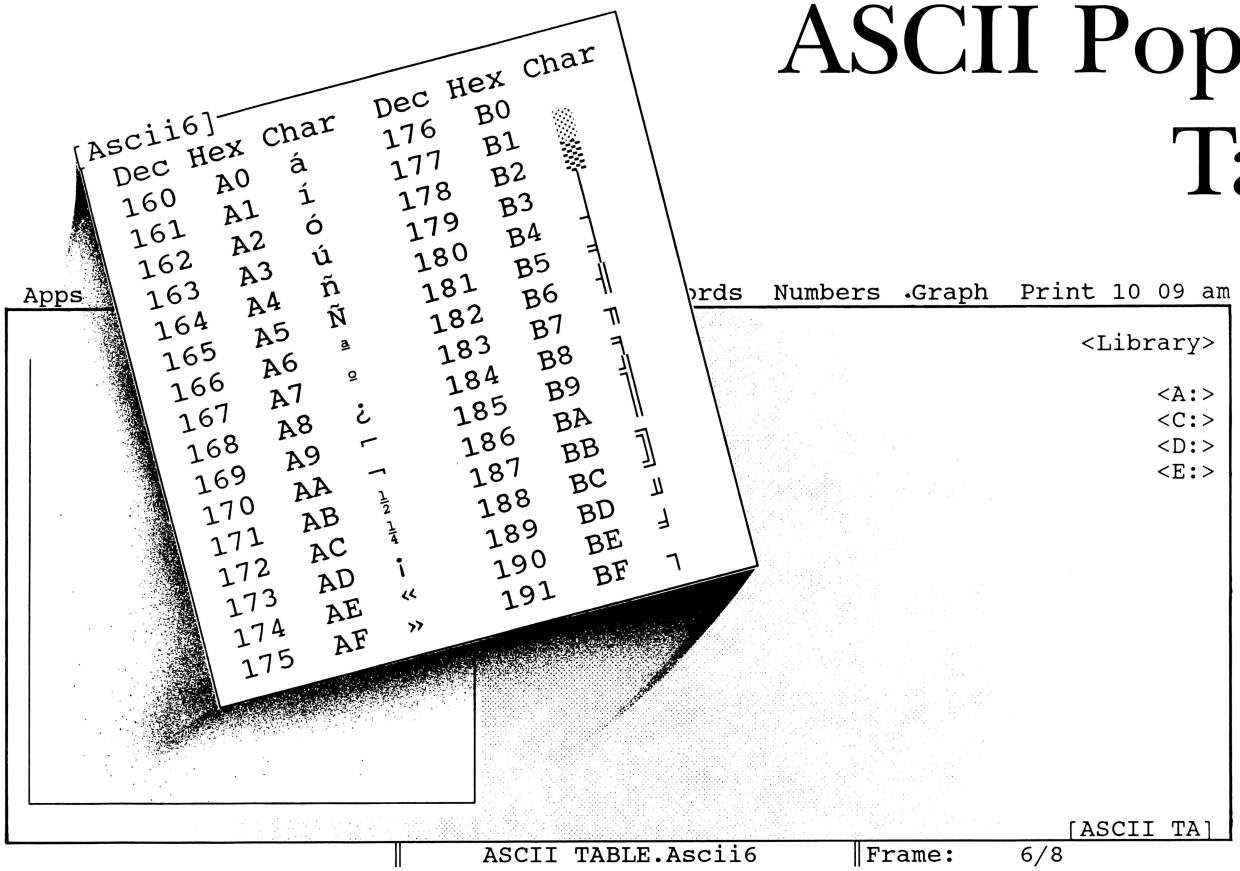
## Documentation Index

The index for the second edition of the documentation contains incorrect page references. The specific references are shown in Figure 2.

| Item | Incorrect | Correct |
|---|---|---|
| Function keys (Page X-17) | U5-224 - U5-225 | U5-234 |
| Memo field (Page X-20) | | |
| adjusting width for output | U5-234 | U5-243 |
| editing in programs | P5-16 | P5-26 |

Figure 2

# ASCII Pop-Up Table




By Kenneth N. Getz

The IBM PC extended character set includes foreign characters, line- and box-drawing characters, and mathematical symbols. You can embed most of these characters in your Framework II documents by pressing **Alt-<n>**, where <n> is the ASCII code (from the numeric keypad) that corresponds to the character you want. This works for every character except the ASCII codes 0, 9, 13, and 255 because these are nonprintable characters in Framework II. This method has one drawback—most of us need to look up the ASCII codes each time we want to use one.

Wouldn't it be nice to have a pop-up chart that shows the ASCII codes and the characters? Wouldn't it be even nicer if you could call it up from Framework II by pressing **Alt-A**? The programs presented in this article, Chart.FW2 and {Alt-A}.FW2, provide just that. From this chart you can select a character to embed by simply highlighting the character and pressing **Return**. Chart creates the ASCII table and places it in your Library Cabinet. {Alt-A} is a macro that brings the ASCII table to the desktop. Whether you call {Alt-A} from the desktop or from within a document, the ASCII table always displays on the desktop in a separate stand-alone frame. If you are in a zoomed document when you press {Alt-A}, the document is forced back into desktop mode. Figure 1 shows the default mode of the ASCII chart on the desktop.

When the ASCII table is in a default mode, it displays in the upper left corner of the screen and the first page of eight is shown. Once the table is displayed on the desktop, you can use **F3 (DRAG)** to place it in a different location. **F3** is the only function key that you can use when the ASCII table is on the desktop; all other function keys are disabled. Unless you press **Esc** to exit, this new location becomes the default location of the table. This is also true for the current page of the table; the page displaying at the time you exit the table becomes the default page. Pressing **Esc** to exit resets both the screen position and the page number to their original, default values.

| Keystroke | Result |
|---|---|
| **Uplevel** | From inside the table, this keystroke moves the cursor to the frame border.<br>From the table frame border, this keystroke has no effect. |
| **PgUp,**<br>**PgDn** | Works the same way whether you are on the table frame border or inside the table. **PgUp** moves the cursor to the frame border and then scrolls to the previous page of the table. **PgDn** moves the cursor to the frame border and then scrolls to the next page of the ASCII table. |
| **Uparrow,**<br>**Dnarrow** | From inside the table, the keystrokes **Uparrow** and **Dnarrow** move vertically through the table, one character at a time. When you reach the top or bottom of a page, the cursor moves to the adjacent page.<br>From the table frame border, these work the same as PgUp and PgDn. |
| **Leftarrow,**<br>**Rightarrow** | From inside the table, these keystrokes move the cursor from one column to the other.<br>From the table frame border, these have no effect. |
| **Home** | From inside the table, **Home** moves the cursor to the top of the current column.<br>From the table frame border, this keystroke moves the cursor to the first page of the table. |
| **End** | From inside the table, **End** moves the cursor to the bottom of the current page.<br>From the table frame border, this keystroke moves the cursor to the last page of the table. |

Table 1

| Keystroke | Result |
|---|---|
| **Esc** | Removes the table from the desktop and resets the page number and screen position to the default values. When you call the table again, it displays in the upper left corner of the screen and is positioned on page 1. |
| **Return** | From inside the table, this keystroke places the character you selected at the current cursor position in your document, then removes the table from the desktop, saving the current table location, page number, and cursor position within the table. When you call the table up again, it displays in the same location on the screen with the cursor positioned at the last character you chose.<br><br>From the table frame border, this keystroke removes the table from the desktop and saves the current table location and page number. The next time you use the table, it displays in that location and on that page number. |

Table 2

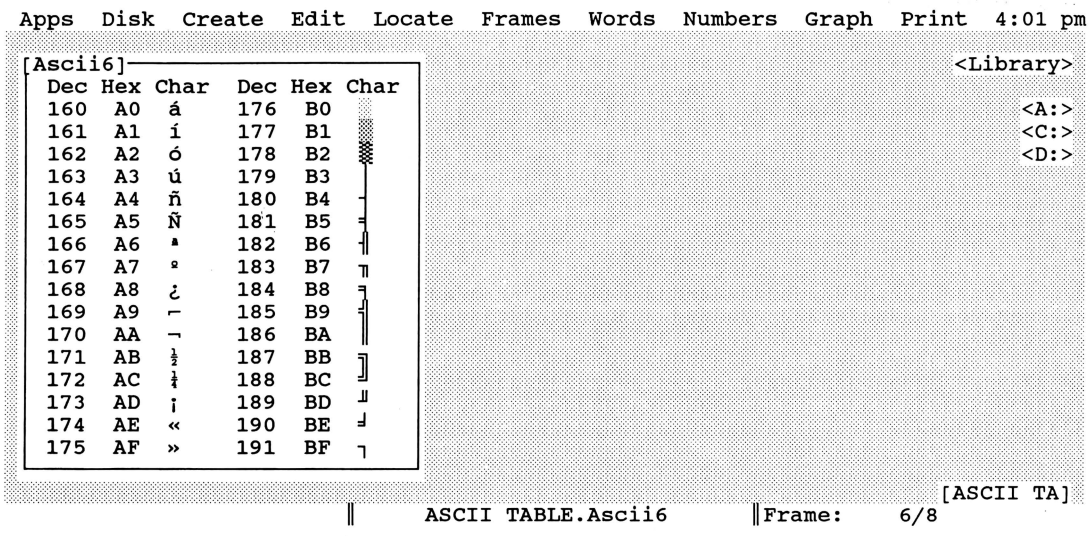



Figure 1

There are several ways to navigate through the ASCII table. Table 1 shows a list of the available navigational keystrokes and their results.

To place a character in a Framework II document (a database, spreadsheet, or word frame), simply highlight the character in the ASCII table and press **Return**. Each time you select a character from the ASCII table, the character is placed in your document at the current cursor location. Then the table is saved back to the Library with the current page number and screen location until you press **Alt-A** again.

There are two ways to exit from the ASCII table: press **Esc** or **Return**. Table 2 shows how these exiting keystrokes behave in this program.

Once the ASCII table is displayed, you can **DRAG (F3)** the frame to any position on the screen. If you drag the table partially off the screen and save it that way, the program will readjust the chart to fit on the desktop the next time you call it up.

## Setting Up the Programs

Both of the programs (Chart and {Alt-A}) contain several subframes. The subframes in the Chart program contain subroutines that are called by Chart.Makeframe to create the ASCII table. You need to run Chart only once to create the ASCII table. After the table is created and saved in the Library, you execute the {Alt-A} macro to display the ASCII table. The subframes in {Alt-A} contain functions, key filters, and global variables to place the ASCII table on the screen and to define the available keystrokes.

**Note**: Sizeto, the user-defined function that creates a word frame based on the size and location you specify, allows you to create customized frames under program control. For details on how to use this function in other programs, refer to Framework Tips in the June 1987 issue of ***TechNotes***.

The subframe labeled Backtab is included as an escape path so that, when you enter the program code and test it for the first time, you can always get back to the default keyboard configuration if an error occurs. If an error does occur during program execution, press **Shift-Tab** to return to the default keyboard configuration.

### {Alt-A}

To set up {Alt-A}, go into the Macros section of the

Library, create an empty word frame, and label it {Alt-A}. (Be sure to include the curly braces, {}, because they tell Framework II that it's a macro frame.) Inside of this frame, create 28 empty word frames and label them using the frame names shown in the list below:

```
{Alt-A}
   - Main
   - Setkeys
   - Getpage
   - Sizeto
   - F3
   - Escape
   - Return
   - Uplevel
   - Dnlevel
   - Home
   - End
   - Left
   - Right
   - Up
   - Down
   - Pgup
   - Pgdn
   - Null
   - Backtab
   - Max rows
   - Max cols
   - Last row
   - Last col
   - Old row
   - Old col
   - Old frame
   - Old view
   - Old page
```

Enter the corresponding FRED code found at the end of this article into the formula area of the containing frame ({Alt-A}) and the formula areas of the first 19 subframes (Main to Backtab). The remaining frames are used as global variables and some need to have initial values. Table 3 contains a list of the frame names with a short description of each variable and the initial value. Place the values listed in Table 3 into the formula area of the respective frames. Once the program code is entered and the global variables initialized, save this macro in the Library with **Alt-F2**.

## Chart

To set up Chart, create an empty word frame on the desktop and label it Chart. Then create five empty word frames inside of Chart and label them.

```
Chart
  - Makeframe
  - Decchar
  - Fillchart
  - Hex
  - Pad
```

In the formula area of the containing frame and each of the five internal frames, enter the corresponding formulas found at the end of the article.

When you're ready to create the ASCII table, verify that {Alt-A} is in the Library Cabinet, then load Chart onto the desktop, highlight the border and press **F5 (RECALC)**. {Alt-A} must be available for Chart to use because Chart calls the Sizeto function to create and place the frame. If the ASCII chart already exists in the Library, the program will return the message "ASCII TABLE already exists" when you attempt to execute Chart again. Press {Alt-A} at any time to call up the ASCII table.

| Frame Name | Use | Initial Value |
|---|---|---|
| Last row | Row where table was last displayed | 1 |
| Last col | Column where the table was last displayed | 1 |
| Old row | Row inside table where cursor was left | 0 |
| Old col | Column inside table where cursor was left | 0 |
| Old page | Page of the table last displayed | 1 |

Table 3

## Chart

```
; Chart.FW2
;   by Kenneth N. Getz
;   November 1, 1987
;
@Chart.Makeframe
```

## Chart.Makeframe

```
; @Makeframe
;
; Controlling frame for preparing the ASCII table.
; Make sure that {Alt-A} has been typed in and is currently
; in the Library before trying to RECALC (F5)  this program.
;
@local(fname, i, mac),
;
; Create the containing frame and turn off borders and labels.
@echo(#OFF),
@isabend(#FALSE),
@performkeys("{Ctrl-L}GLibrary.ASCII TABLE{Return}"),
@if(@not(@isabend),
     @list(
          @echo(#ON),
          @prompt("ASCII TABLE already exists.  Chart exiting."),
          @return(0)
     )
),
@SizeTo("ASCII TABLE", 1, 1, 21, 32),
@performkeys("{Ctrl-F}H{Ctrl-F}D"),
;
; Create 8 internal frames for the actual pages of the table and then
; fill them with the ASCII data.
i := 1,
@while(i <= 8,
     @echo(#OFF),
     fname := "Ascii" & @integer(i),
     mac := "@SizeTo(""" & fname & """, 2, 2, 20, 31)",
     @mac,
     @echo(#ON),
     @Fillchart(i),
     i := i + 1,
),
;
; Move the whole frame into the Library.
@echo(#OFF),
@setselection("Ascii1"),
@performkeys("{F6}{Ctrl-7}{Dnarrow}{Return}{F7}"),
@setselection("Ascii Table"),
@performkeys("{Dnlevel}{Return}{Ctrl-F}A{Dnarrow}{Uplevel}{F8}"),
@setselection("Library"),
@performkeys("{Dnlevel}{Return}{Uplevel}{Scroll-Lock}{Del}"),
@echo(#ON),
@prompt("Press Alt-A to display ASCII TABLE."),
@return(0)
```

## Chart.Decchar

```
; @Decchar(charnum, col)
;
; Creates the three-part string for each character--the number, the
; hex value of the number, and the corresponding character.
;
@local(expr, string, char),
;
; Characters 0, 9, and 10 cannot be printed, so must substitute a space.
; Character 123 ("{") must be treated specially.
@if(@or(@item1 = 0, @item1 = 9, @item1 = 13),
     char := " ",
     @if(@item1 = 123,
          char := "{{}",
          char := @chr(@item1)
     )
),
;
; Create the display string for this item.  If it's in column 1,
; append 3 spaces on the end.
string := @Pad(@item1, 4) & @Hex(@item1) & @rept(" ", 2) & char &
               @if(@item2 = 1,
                    @rept(" ", 3)
               ),
@return(string)
```

## Chart.Fillchart

```
; @Fillchar(pagenumber)
;
; Creates a full page for each of the 8 pages.
;
@local(newline, start, i, header),
header := " Dec Hex Char",
; The following line assigns a single CR to the variable newline.
; There are no spaces after the quote.
newline := "
",
@performkeys("{Dnlevel}{Ctrl-Home}" & header & " " & header),
;
; Get the number of the first character on the page.
start := (@item1 - 1) * 32,
i := 0,
;
; Do all 16 lines per page.
@while(i < 16,
     @performkeys(newline & @Decchar(start + i, 1) & @Decchar(start + i + 16, 2)),
     i := i + 1
)
```

## Chart.Hex

```
; @Hex(2-digit number)
;
; A decimal-to-hex conversion, simple since it is assured no number larger
; than 255.
;
; This routine runs much faster if the "Keep Overlays in Memory" is set ON
; in Framework II Setup.
;
@local(expr, string, digits),
```

```
digits := "0123456789ABCDEF",
;
; If the number is less than 16, put a space in front.  If not, the first
; digit is the number DIV 16, and the second digit is the number MOD 16.
expr := @if(@item1/16 < 1,
                " ",
                @mid(digits, @floor(@item1/16) + 1, 1)
          )
          & @mid(digits, @mod(@item1, 16) + 1, 1),
string := @rept(" ", 4 - @len(expr)) & expr,
@return(string)
```

## Chart.Pad

```
; @Pad(expr,width)
;
; Used to pad an expression to a specific width with leading spaces.
;
@return(@rept(" ", @item2 - @len(@integer(@item1))) & @integer(@item1))
```

## {Alt-A}

```
; [{Alt-A}].FW2
; November 1, 1987
; By Kenneth N. Getz
;
; Controls the display and manipulation of the ASCII table stored in the
; library by the program Chart.  This program, however, must be in place
; before Chart is run, and cannot be run until Chart has created the table.
;
@[{Alt-A}].Main
```

## [{Alt-A}].Main

```
; ASCII Table Display Program
;
; Dnlevel, Leftarrow and Rightarrow let you go into the frames and choose a
; character to place into the current text.
;
; PgUp and PgDn (pressed at any time) and Uparrow and Dnarrow (pressed from
; the frame border) let you page through all the ASCII table pages.
;
; Return from within the frame places the chosen character into the current
; frame and saves the current ASCII table screen position. Return from the
; frame border removes the table but remembers what page you were on.  It
; also remembers where on the screen the ASCII table was left.
;
; Uplevel from within the frame just moves you to the frame border.
;
; Esc pressed at any time removes the table and resets the page back to
; page 1.  It also resets the screen position of the table to 1,1.
;
; F3 can be used to DRAG the frame around the desktop.
;
@eraseprompt,
@prompt("Preparing ASCII table..."),
@echo(#OFF),
Old view := @frametype,
;
; Store away where the table is called from.
@performkeys("{Uplevel}"),
Old frame := @setselection,
@if(@scan("[]", Old frame, 1) <> #N/A!,
```

```
     @list(
          @echo(#ON),
          @prompt("Please name all frames before calling ASCII TABLE."),
          @return(0)
     )
),
@if(@iserr(@setselection("Library.ASCII TABLE")),
    @list(
         @echo(#ON),
         @beep,
         @prompt("You must create ASCII TABLE by running Chart first."),
         @return(0)
    )
),
@Sizeto("ASCII TABLE", last row, last col, last row + 20, last col + 31),
@performkeys("{Ctrl-F}H{Ctrl-F}D" &
                "{Ctrl-L}GLibrary.ASCII TABLE.Ascii1{Return}" &
                "{F6}{Ctrl-7}{Dnarrow}{Return}" &
                "{F8}{Ctrl-Uplevel}{Scroll-Lock}"),
@performkeys("{Dnlevel}{Return}{Ctrl-F}A{Dnarrow}" &
                "{Ctrl-Uplevel}{Ctrl-F}O{Dnlevel}" &
                "{Scroll-lock}{Return}{Scroll-lock}"),
;
; Set all the keyfilters.
@[{Alt-A}].Setkeys,
;
; Find the correct starting page for display.
@if(Old page <> 1,
     @performkeys(@rept("{Dnarrow}", Old page - 1))
),
; If we need to go into the frame, do it.
@if(@and(Old row <> 0, Old col <> 0),
     @Dnlevel
),
; And finally display the frame.
@echo(#ON),
@eraseprompt
```

## [{Alt-A}].Setkeys

```
; Setkeys
;
; Sets up all necessary keyfilters.
;
@keyfilter({All}, Null),
@keyfilter({Backtab}, Backtab),
@keyfilter({Dnarrow}, Down),
@keyfilter({Dnlevel}, Dnlevel),
@keyfilter({End}, End),
@keyfilter({Esc}, Escape),
@keyfilter({F3}, F3),
@keyfilter({Home}, Home),
@keyfilter({Leftarrow}, Left),
@keyfilter({Pgdn}, Pgdn),
@keyfilter({Pgup}, Pgup),
@keyfilter({Return}, Return),
@keyfilter({Rightarrow}, Right),
@keyfilter({Uparrow}, Up),
@keyfilter({Uplevel}, Uplevel)
```

## [{Alt-A}].Getpage

```
; Getpage
;
; Returns the page number of the current table page.
; Assumes the heading to be in format "xxxxxxx#", where # is the page number.
; Also assumes that the frame border is highlighted.
;
@return(@value(@mid(@setselection, @len(@setselection), 1)))
```

## [{Alt-A}].Sizeto

```
; Sizeto
;
; Used to create a frame at a specified location on the screen.
;
; @Sizeto(<frame name>, <upper left row>, <upper left col>,
;                   <lower right row>, <lower right col>)
;
; Returns 0 if frame formed OK, -1 if there as an error.
;
; Make sure we're at the desktop level.
@performkeys("{Ctrl-Uplevel}"),
;
; Create the new frame and zoom to full screen, storing away the maximum
; screen coordinates.  Verify that the parameters passed conform to the
; possible position values.
;
@performkeys("{Ctrl-C}E{F4}{F4}"),
Max rows := @panel2,
Max cols := @panel1,
@if(@or(@item2 < 1, @item3 < 1, @item4 > Max Rows + 2, @item5 > Max Cols + 2),
      @list(
            @performkeys("{Return}{Del}"),
            @return(-1)
      )
),
; Perform DRAG (F3) to move upper left corner.
@performkeys("{F3}"),
@performkeys(@rept("{Ctrl-Dnarrow}", @int((@item2 - 1)/4)) &
                  @rept("{Dnarrow}", @mod(@item2 - 1, 4)) &
                  @rept("{Ctrl-Rightarrow}", @int((@item3 - 1)/8)) &
                  @rept("{Rightarrow}", @mod(@item3 - 1, 8)) &
                  "{Return}"
),
;
; Perform SIZE (F4) to move lower right corner.
@performkeys("{F4}" &
                  @rept("{Ctrl-Uparrow}", @int((Max rows + 2 - @item4)/4)) &
                  @rept("{Uparrow}", @mod(Max rows + 2 - @item4, 4)) &
                  @rept("{Ctrl-Leftarrow}", @int((Max cols + 2 - @item5)/8)) &
                  @rept("{Leftarrow}", @mod((Max cols + 2 - @item5), 8)) &
                  "{Return}"
),
;
; Put the name in.
@performkeys(@item1 & "{Return}"),
@return(0)
```

## [{Alt-A}].F3

```
; F3
;
; Used to DRAG the ASCII table around the screen.  Allows the use of all the
; arrow keys, and stops processing when you press Return.
;
; First find out if we are in the table or on the border.
@local(view),
view := @frametype,
;
; Set the necessary keys back to their original use.
@keyfilter({Leftarrow}),
@keyfilter({Rightarrow}),
@keyfilter({Uparrow}),
@keyfilter({Dnarrow}),
@keyfilter({Ctrl-Leftarrow}),
@keyfilter({Ctrl-Rightarrow}),
@keyfilter({Ctrl-Uparrow}),
@keyfilter({Ctrl-Dnarrow}),
@keyfilter({Tab}),
@keyfilter({Backtab}),
@keyfilter({Return}),
;
; We have to disable F3 and Esc while performing the DRAG.
@keyfilter({F3}, Null),
@keyfilter({Esc}, Null),

; Move to the frame border and set up for DRAG.  Keep processing
; keys until a Return is pressed.
@performkeys("{Ctrl-Uplevel}{F3}"),
@while(@nextkey <> {Return},
     @performkeys(@keyname(@key))
),
@performkeys("{Return}"),
;
; Go back inside the frame.  If we were inside the frame before, we
; have to use Ctrl-Dnlevel to get back where we were.
@if(view = 14,
     @performkeys("{Ctrl-Dnlevel}"),
     @performkeys("{Dnlevel}")
),
;
; Put the keyfilters back for the rest of the program.
@keyfilter({Leftarrow}, Left),
@keyfilter({Rightarrow}, Right),
@keyfilter({Uparrow}, Up),
@keyfilter({Dnarrow}, Down),
@keyfilter({Ctrl-Leftarrow}, Null),
@keyfilter({Ctrl-Rightarrow}, Null),
@keyfilter({Ctrl-Uparrow}, Null),
@keyfilter({Ctrl-Dnarrow}, Null),
@keyfilter({Tab}, Null),
@keyfilter({Backtab}, Backtab),
@keyfilter({Return}, Return),
@keyfilter({F3}, F3),
@keyfilter({Esc}, Escape)
```

## [{Alt-A}].Escape

```
; Escape
;   @Escape(@item1)
;
;   If @item1 = #TRUE,  quit saving the cursor position, page number, and
;                            screen position of the table.
;   If @item1 <> #TRUE, quit without saving anything, and reset all values
;                       to their defaults.
;
@local(old),
;
; If we're ESCaping and not saving anything, reset all values back to
; their defaults.
@if(@isna(@item1),
     @list(
          Old page := 1,
          Old row  := 0,
          Old col  := 0,
          Last row := 1,
          Last col := 1
     )
),
;
; Get to the outer frame border.
@performkeys("{Ctrl-Uplevel}"),
;
; If necessary, get the current screen position of the table.
@if(@item1 = #TRUE,
     @list(
          ; Get the current screen coordinates, and adjust if not reasonable.
          @performkeys("{F3}"),
          Last row := @panel2,
          Last col := @panel1,
          @if (Last row = 0,
               Last row := 1,
               @if (Last row > Max rows - 18,
                    Last row := Max rows - 18
               )
          ),
          @if (Last col = 0,
               Last col := 1,
               @if (Last col > Max cols - 29,
                    Last col := Max cols - 29
               )
          ),
          @performkeys("{Return}")
     )
),
;
; Delete the current frame.
@performkeys("{Del}"),
;
; Go back to the original frame.
old := @Old frame,
@setselection(old),
;
; If we were inside the frame, go in.
@if (Old view >= 11,
```

```
     @performkeys("{Dnlevel}")
),
; Reset all the keyfilters.
@keyfilter({All})
```

## [{Alt-A}].Return

```
; Return
;
; Pressing Return puts away the ASCII table and, if the cursor was on the
; frame border, acts as if Escape was pressed.  If it was inside the frame,
; removes the table, stores the current position of the table, stores the
; current position of the cursor, and places the chosen character in the
; calling frame.
;
@local(old, char, inframe),
;
; Test to see if we're inside the frame or not.
; If so, Uplevel out of it.
inframe := (@frametype = 14),
@if(inframe,
     @list(
          ; Save the current character, cursor row and column positions.
          char    := @textselection,
          Old row := @panel2,
          Old col := @panel1,
          @performkeys("{Uplevel}")
     )
),
Old page := @Getpage,
@Escape(#TRUE),
;
; If we were inside a frame, put the chosen character in the calling frame.
@if(inframe,
     @performkeys(char)
),
; Clear out all the keyfilters.
@keyfilter({All})
```

## [{Alt-A}].Uplevel

```
; Uplevel
;
; If we're on the frame border, do nothing.
; If we're currently inside a word frame, reset cursor coordinates
; and move to the border.
;
@if(@frametype = 14,
     @list(
          Old row := 0,
          Old col := 0,
          @performkeys("{Uplevel}")
     )
)
```

## [{Alt-A}].Dnlevel

```
; Dnlevel
; If we're inside a word frame, do nothing.
; If we're on the frame border, then either go to the first character on the
;   page or to the last chosen character.
;
```

```
@local(page),
;
; If we're inside the frame, do nothing.
@if(@frametype = 14,
     @return(0)
),
;
; We must be on the border, so see what page we're on and then go in.
page := @Getpage,
@performkeys("{Dnlevel}{Ctrl-home}"),
@if(@or(@and(Old col = 0, Old row = 0), Old page <> page),
     @performkeys(@rept("{Rightarrow}", 10) & "{Dnarrow}"),
     @performkeys(@rept("{Rightarrow}", Old col - 1) &
                  @rept("{Dnarrow}", Old row-1))
)
```

## [{Alt-A}].Home

```
; Home
;
; If we're inside a frame, Home takes the cursor to the top of the current
; column.  If on the frame border, Home takes us to Page 1 of the table.
;
@local(row, frame),
@if(@frametype = 14,
     @list(
          row := @panel2,
          @performkeys(@rept("{Uparrow}", row - 2))
     ),
     @list(
          frame := @panel1,
          @if(frame <> 1,
               @list(
                    @echo(#OFF),
                    @performkeys(@rept("{Dnarrow}", 9 - @Getpage)),
                    @echo(#ON)
               )
          )
     )
)
```

## [{Alt-A}].End

```
; End
;
; If we're inside a frame, End takes the cursor to the bottom of the current
; column.  If on the frame border, End takes us to Page 8 of the table.
;
@local(row, frame),
@if(@frametype = 14,
     @list(
          row := @panel2,
          @performkeys(@rept("{Dnarrow}", 17 - row)),
     ),
     @list(
          frame := @panel1,
          @if(frame <> 8,
               @list(
```

```
                    @echo(#OFF),
                    @performkeys(@rept("{Dnarrow}", 8 - @Getpage)),
                    @echo(#ON)
               )
          )
     )
)
```

## [{Alt-A}].Left

```
; Left
;
; If we're not inside a word frame, do nothing.
; If inside a word frame, then either go left 14 columns or go right 14
; columns, depending on whether the cursor is on the left column of
; characters or the right column of characters.
;
@if(@frametype = 14,
     @if(@panel1 = 25,
          @performkeys(@rept("{Leftarrow}", 14)),
          @performkeys(@rept("{Rightarrow}", 14))
     )
)
```

## [{Alt-A}].Right

```
; Right
;
; If we're not inside a word frame, do nothing.
; If inside a word frame, then either go left 14 columns or go right 14
; columns, depending on whether the cursor is on the left column of
; characters or the right column of characters.
;
@if(@frametype = 14,
     @if(@panel1 = 11,
          @performkeys(@rept("{Rightarrow}", 14)),
          @performkeys(@rept("{Leftarrow}", 14))
     )
)
```

## [{Alt-A}].Up

```
; Up
;
; If we're inside a word frame and on the second row, go to the bottom row.
; Otherwise, just perform the Uparrow.  If inside the frame, this goes up
; one character.  If on the frame border, this goes up one page.
;
@if(@and(@frametype = 14, @panel2 = 2),
     @performkeys(@rept("{Dnarrow}", 15)),
     @performkeys("{Uparrow}")
)
```

## [{Alt-A}].Down

```
; Down
;
; If we're inside a word frame and on the bottom row, go to the first row
; of characters.  Otherwise, just perform the Dnarrow.  If inside the frame,
; this goes down one character.  If on the frame border, this goes
; down one page.
;
@if(@and(@frametype = 14, @panel2 = 17),
```

```
    @performkeys(@rept("{Uparrow}", 15)),
    @performkeys("{Dnarrow}")
)
```

## [{Alt-A}].PgUp

```
; PgUp
;
; No matter where we are, this moves to the previous display page.
; If inside a word frame, this necessitates getting to the frame
; border first.  This also resets the row and column counters.
;
@if(@frametype = 14,
          @performkeys("{Uplevel}")
),
@performkeys("{Uparrow}"),
Old page := @if(old page = 1,
                    8,
                    old page - 1
                )
```

## [{Alt-A}].PgDn

```
; PgDn
;
; No matter where we are, this moves to the next display page.
; If inside a word frame, this necessitates getting to the frame
; border first.  This also resets the row and column counters.
;
@if(@frametype = 14,
       @performkeys("{Uplevel}")
),
@performkeys("{Dnarrow}"),
Old page := @if(Old page = 8,
                1,
                Old page + 1
                )
```

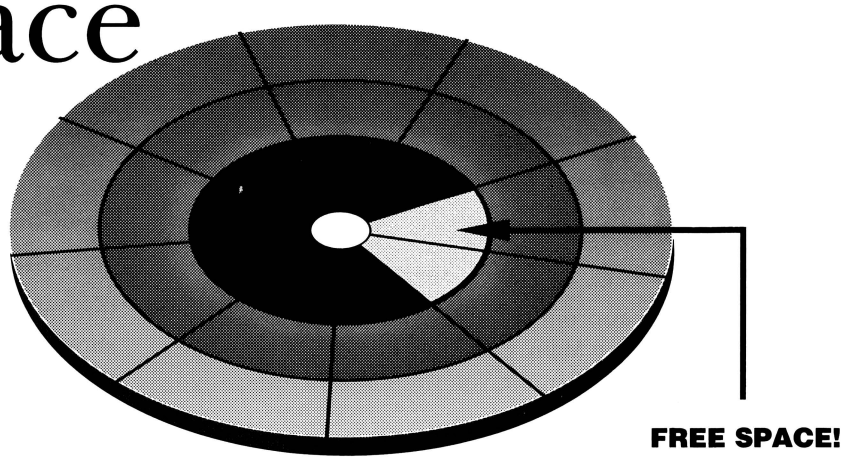
## [{Alt-A}].Null

```
; Null
;
; A do-nothing frame.
;
@return(0)
```

## [{Alt-A}].Backtab

```
; Backtab
;
@keyfilter({All})
```

# Finding Free Disk Space




By Kenneth N. Getz

The program presented here, Dskspace.FW2, will allow you to determine from within Framework II how much free disk space you have on any disk in your system. Besides being a useful utility, this program shows how you can use the *Framework II Developer's Toolkit* to add "hooks" to DOS routines that would otherwise be unavailable from within Framework II. The *Framework II Developer's Toolkit* contains, among other things, assembly language and C calls to access Framework II's low-level interface. These calls were modified for this program to allow access to some information that DOS provides.

The program works by calling DOS interrupt number 21H, service 36H, to get free disk space. This function needs to know the drive number in question (0 for the current drive, otherwise 1 = A:, 2 = B:, and so on) and returns either FFFFh if you specified an invalid drive number, or four pieces of information if the drive is legal. If the specified drive is valid, this DOS call returns the following values in the indicated registers:

AX = sectors per cluster
BX = number of available clusters
CX = bytes per sector
DX = clusters per drive

To find the actual amount of free disk space, we return sectors per cluster, bytes per sector, and available clusters, and let Framework II multiply them together to find the disk space.

This program was created and tested with Microsoft C, version 4.0, and with Borland International's Turbo C, version 1.0. To create the executable version of this program, you will need either of those compilers, Microsoft MASM version 3.0 or greater, and the *Framework II Developer's Toolkit.* Or, you can download the program by modem as described in On-Line Services.

## Setup

First, you will need to make some changes to Mymain.ASM from the \LLI\RDTXT directory of your Framework II Developer's Toolkit Disk #2, as described in the article "Using the *Framework II Developer's Toolkit* with Microsoft C" in the June 1987 issue of ***TechNotes***. This includes PUSHing and POPping SI and DI in each of the C-callable routines in the second portion of the file. (If you don't have the June 1987 issue, you can order it by calling 1-800-828-2514 or 1-800-331-4164 from California.) You will need to make a copy of Mymain.ASM, call it Diskmain.ASM, and modify the first portion as shown in the code listings in this article.

Dskspace requires two external programs, Disk.C and Getfree.ASM. You should create these files by typing in the source code listings at the end of the article and saving them to disk as text files.

You should also create a Framework II word frame named Dskspace with five internal frames so that the structure looks like this (the comments explain what will be in each frame):

```
-Dskspace
     - Main
     - Drive   ; Drive number (0 - max)
     - Avail   ; Number of available clusters
     - Bsec    ; Bytes per sector
     - Sclus   ; Clusters per drive
```

Type the appropriate FRED code into Dskspace and Dskspace.Main. Leave the other frames empty for now. At this point, you should have four files:

Diskmain.ASM
Disk.C
Getfree.ASM
Dskspace.FW2

## Compiling

Before compiling and/or assembling these files (Figure 1), all assemblers and compilers must be installed according to the instruction in their manual. Make sure at this point that all programs have been installed correctly and are either in the current directory or in the DOS path.

## Assembling the Framework II File

Once you have created Getfree.EXE, you are ready to complete the frame. Load Dskspace.FW2 to the desktop, go inside it, and press **{Ctrl-D}G** to get the file Getfree.EXE by name and load it into the frame.

To test your new function, create another word frame on the desktop and enter the formula

```
@Dskspace(0)
```

into its formula area. **RECALC (F5)** this frame. You should see the free disk space on the current drive displayed in its contents area.

You can use the techniques presented here to access many of the DOS or BIOS interrupt services from within Framework II. This process is simplified by the fact that Framework II handles critical errors itself (such as writing to a disk that isn't ready or printing to a printer that isn't on-line). However, be sure to test rigorously any modules you write in this way because when programs written with the low-level interface crash they will most often crash the entire system.

| **Command** | **Type** |
|---|---|
| *To assemble* | |
| Getfree.ASM | `MASM Getfree;` |
| Diskmain.ASM | `MASM Diskmain;` |
| *To compile Disk.C* | |
| Microsoft C | `MSC Disk /Gs;`<br>(The /Gs disables stack checking.) |
| Turbo C | `TCC -c Disk`<br>(Turbo uses no stack checking by default.) |
| *To link all the modules* | |
| Microsoft C | `LINK Diskmain+Disk+Getfree, Getfree /NOD;`<br>(The /NOD leaves out the default libraries.) |
| Turbo C | `TLINK Diskmain Disk Getfree, Getfree`<br>(Turbo leaves out default libraries unless told otherwise.) |

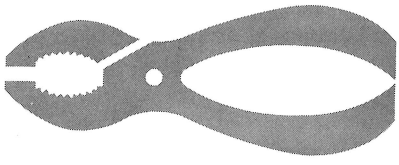
Figure 1

## Diskmain.ASM

```
;
; This example contains only the header portion of the file.
; You must refer to the file Mymain.ASM in the \LLI\RDTXT directory
; of your Disk #2 of the Framework II Developer's Toolkit and to the
; article "Using the Framework II Developer's Toolkit with Microsoft C" in
; the June 1987 issue of TechNotes.
;
        extrn   _diskspace:near

_DATA   SEGMENT WORD    PUBLIC  'DATA'
_DATA   ENDS
CONST   SEGMENT WORD    PUBLIC  'CONST'
CONST   ENDS
_BSS    SEGMENT WORD    PUBLIC  'BSS'
_BSS    ENDS

DGROUP  GROUP   CONST, _BSS, _DATA,_STACK
_DATA   segment word PUBLIC 'DATA'
PUBLIC FW_EXEC
fw_exec dd      0               ; Will be plugged by Framework II.
proc1   dw      offset getfreef ; Offset addr of proc #1.
dummy_buffer    dw      0       ; Used by num_bytes.
public  __acrtused              ; Trick to force in startup.
__acrtused = 9876h              ; Funny value not easily matched in SYMDEB.
                                ; Not really necessary for Turbo C.
_DATA   ends

_STACK  segment word stack 'DATA'
        dw 1024 dup (?)         ; Declare stack for local usage
_STACK  ends

; Segmentation set up to reflect the SMALL model as implemented
; by various C compilers

PGROUP  GROUP _TEXT
_TEXT   SEGMENT BYTE PUBLIC 'CODE'
ASSUME CS:_TEXT,DS:DGROUP,ES:DGROUP,SS:DGROUP

; The following procedure is the FAR entry point. Its
; only purpose is to call the real (near) routine which is
; written in C.

getfreef        proc    far
        call    _diskspace
        ret
getfreef        endp



; The following routines are C-callable interfaces to the
; Framework II service routines. The comment header on each
; one indicates how the procedure would look to the calling
; C program

; The rest of the code is unchanged and can be used as is once it is
; modified as described in the June TechNotes article.
```

## Disk.C

```
/*  File.....: Disk.C                                                    */
/*  Author...: Kenneth N. Getz                                           */
/*  Date.....: November 1, 1987                                          */
/*  Notice...: 1987, All rights reserved.                                */
/*  Notes....: Disk.C returns the amount of disk space free to FWII.     */
/*             Turbo C actually includes the function Getdfree() in its  */
/*             default library.  You could include that here instead of  */
/*             using Getfree.ASM, but that does complicate the linking   */
/*             process and forces the inclusion of the default libraries,*/
/*             which the writers of the Toolkit have advised against     */

struct dfree {
  unsigned df_avail;
  unsigned df_total;
  unsigned df_bsec;
  unsigned df_sclus;
  };

struct data {
  int type;
  union {
       int integer;        /* Storage for the actual value in question. */
       char bcd[10];     /* Just space to fill out the union to 10 bytes. */
       } value;
  };

void getfree(int, struct dfree *);

int diskspace()
{

   struct dfree answer;
   struct data value_rec;

   GET_VAL(1,&value_rec);                        /* Get the drive from FWII. */
   if (value_rec.type != 3)                       /* If it's not an integer, */
      return(value_rec.type);              /* send back what it was and quit. */
   getfree(value_rec.value.integer, &answer);           /* Get free space. */
   value_rec.value.integer = answer.df_avail;        /* Available clusters. */
   PUT_VAL(2,&value_rec);                            /* Send back to FWII. */
   value_rec.value.integer = answer.df_bsec;         /* Number of sectors. */
   PUT_VAL(3,&value_rec);                            /* Send back to FWII. */
   value_rec.value.integer = answer.df_sclus;       /* Sectors per cluster. */
   PUT_VAL(4,&value_rec);                            /* Send back to FWII. */
   return(0);
}
```



## Getfree.ASM

```
; Program ...: Getfree.ASM
; Author ....: Kenneth N. Getz
; Date ......: November 1, 1987
; Notes .....: Assembly routine to call DOS and return the necessary
;              information.
;
_TEXT   SEGMENT BYTE    PUBLIC  'CODE'
        ASSUME  CS:_TEXT

PUBLIC  _getfree
_getfree        PROC    NEAR
        PUSH    BP                      ; Save the stack pointer.
        MOV     BP,SP                   ; Position BP to reference the stack.
        PUSH    SI                      ; Save SI (to keep C happy.)
        SUB     DX,DX                   ; Clear out DX (drive number.)
        MOV     DX,[BP+4]               ; Get the drive number into DX.
        MOV     AH,36h                  ; Get set for DOS call.
        INT     21h                     ; Call DOS for free space.
        MOV     SI,[BP+6]               ; Get address of structure.
        MOV     [SI+0],BX               ; Move values into structure.
        MOV     [SI+2],DX
        MOV     [SI+4],CX
        MOV     [SI+6],AX
        POP     SI                      ; Restore SI.
        POP     BP                      ; Restore BP.
        RET
_getfree        ENDP

_TEXT   ENDS
END
```

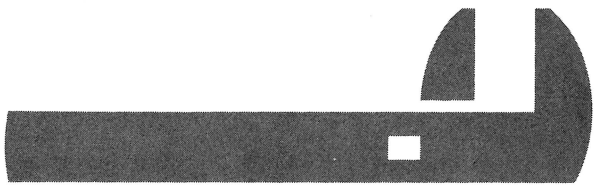
## Dskspace.FW2

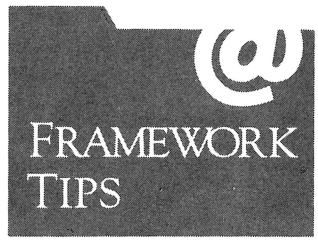
```
; Dskspace.FW2
;
; This passes control to the main program in the internal frame.
;
@Dskspace.main(@item1)
```

## Dskspace.Main

```
; @Dskspace.Main("Drive:")
;
; or
;
; @Dskspace.Main(Drive number)
;
;    by Kenneth N. Getz
;    November 1, 1987
;
;
@local(disk, max),
;
; You can either call this with a drive letter ("A:") or with a drive
; number, which can be any number from 0 (for the current drive), up to
; max (defined below), the maximum number of drives in your system.
;
; This program returns either the amount of free space, if you passed a
; valid drive letter or number as a parameter, or -1, if you send it an
; invalid parameter.
```

```
; Framework II very graciously handles the situation in which you ask for
; the free disk space on a drive that isn't ready (no disk in drive, for
; example), so we don't have to worry about that.
;
; See if the parameter is numeric. If so, only allowed numbers are 0
; (for current default drive) through 1 (for drive A:), 2 for drive
; B:, up to 4 (for D: on my system). This upper limit should be changed
; to match your largest drive.
;
max := 4,
;
; Check to see if the parameter was numeric or character.
@if(@isnumeric(@item1),
  ;
  ; If it was numeric, check to make sure it's valid.
  @list(
       @if(@or(@item1 > max, @item1 < 0),
             @return(-1)
       ),
       ; If valid, store it away for calls to DOS.
       drive := @item1
  ),
  ;
  ; If it was a character string, first convert to uppercase.
  @list(
       disk := @mid(@item1, 1, 1),
       @if(@and(@int(disk) >= 96, @int(disk) <= 122),
             disk := @chr(@int(disk) - 32)
       ),
       ;
       ; Check to make sure the drive letter is valid.
       @if(@not(@and(disk >= "A", disk <= @chr(@int("A") + max - 1))),
             @return(-1)
       ),
       ; Store drive number away for calls to DOS.
       drive := @int(disk) - @int("A") + 1
  )
),
;
; Set up the parameters for the C module and call it.
@execparms("Drive","Avail","Bsec","Sclus"),
@exec(Getdisk,1),
;
; Return the answer.
@return(Avail * Bsec * Sclus)
```






## Conditional @sum

By David Howlett and Quinn Wildman

@Dsum2 is a variation on the user-defined function, @Dsum, found on the Framework II Apps diskette in the file Dsumdcnt.FW2. Both functions calculate the sum of the items in a range for which a condition is true. @Dsum totals the cells in a range for which the cell in a second range is equal to a specified value. @Dsum2 totals the cells in a range for which the cell in a second range falls between two specified values.

@Dsum is called with three parameters:

```
@Dsum(range1, range2, value)
```

Each of the cells in range2 is compared with value. If they are equal, the corresponding cell in range1 is added to the total. The first range must contain numbers. The second range and the value to which it is compared must be of the same data type, whether text, date, or numeric. Dsum2 has four parameters:

```
@Dsum2(range1, range2, low value, high value)
```

Dsum2 walks through range2, comparing the value found there to low value and high value. If the value is greater than or equal to low value and less than or equal to high value, the corresponding cell in range1 is added to the total. The cells in the first range must contain numbers. The values in range2, low value, and high value must all be the same data type. They could be text, dates or, numbers:

```
@Dsum(B2:B7, C2:C7, "Overdue")
```

This formula examines each of the cells in the range C2:C7. If a cell contains "Overdue", the number in the corresponding cell in the range B2:B7 is added to the total.

```
@Dsum2(B2:B7, A2:A7,  @date(1987, 9, 1),
                      @date(1987, 9, 30))
```

Dsum2 examines each of the cells in the range A2:A7. If the value found in the cell is a date between 9/1/87 and 9/30/87 inclusive, the corresponding number in the range B2:B7 is added to the total.

```
; @Dsum2.FW2
; by David Howlett
; Modified by Quinn Wildman
; November 1, 1987
;
; @Dsum2(@item1, @item2, @item3, @item4)
;
; Sums all numeric values in the range @item1 that meet
; the condition that the cell in the range @item2 >= @item3
; and <= @item4.
;
@local(x, inc),
@while((x := @get(@item1)) <> #NULL!,
    @if(@and(@get(@item2) >= @item3, @get(@item2) <= @item4),
        inc := inc + x
    ),
    @next(@item1),
    @next(@item2)
),
@return(inc)
```

## Character to Date Conversion

Incdate.FW2, found in Richard Bunter's article, "Incrementing Dates," in the October issue of ***TechNotes***, can be used to convert a date from one format to another. Ctod.FW2 is a short user-defined function that specifically converts a date in character format to a true Framework II date. Ctod.FW2 might be used with two fields in a database. Field 1 would have a date entered as character, Field 2 would have the formula:

```
Field 2 := @Ctod(Field 1)
```

```
; Ctod.FW2
; by Quinn Wildman
; November 1, 1987
;
; User-defined function returns a true Framework II date given a
; character string in one of four formats:
;
;   1. mm/dd/yy
;   2. mm/d/yy
;   3. m/dd/yy
;   4. m/d/yy
;
; If there are any spaces in the character string, this will not
; work.
;
@if(@item1 = #N/A!,
   @return(" ")
),
@if(@len(@item1) = 8,
       @return(@date(1900 + @value(@mid(@item1, 7, 2)),
                           @value(@mid(@item1, 1, 1)),
                           @value(@mid(@item1, 4, 2))
               )
       ),
       @if(@len(@item1) = 6,
              @return(@date(1900 + @value(@mid(@item1, 5, 2)),
                                   @value(@mid(@item1, 1, 1)),
                                   @value(@mid(@item1, 3, 1))
                      )
              ),
              @if(@mid(@item1, 2, 1) = "/",
                     @return(@date(1900 + @value(@mid(@item1, 6, 2)),
                                          @value(@mid(@item1, 1, 1)),
                                          @value(@mid(@item1, 3, 2))
                             )
                     ),
                     @return(@date(1900 + @value(@mid(@item1, 6, 2)),
                                          @value(@mid(@item1, 1, 2)),
                                          @value(@mid(@item1, 4, 1))
                             )
                     )
              )
       )
)
```

| Default Date | User Enters | @Getdate Result |
|---|---|---|
| November 1, 1987 | {Return} | November 1, 1987 |
| November 1, 1987 | 15 {Return} | November 15, 1987 |
| November 1, 1987 | 1/23 {Return} | January 23, 1987 |
| November 1, 1987 | 6/23/72 {Return} | June 23, 1972 |
| November 1, 1987 | 7/4/1776 {Return} | July 4, 1776 |

Figure 1

## Date Input Function

Getdate.FW2 is another user-defined function for dates. Getdate is written to be used within a FRED program to collect dates, but can be used with a macro as well.

Getdate makes date entry fast and easy. The user is prompted with a default date in the familiar MM/DD/YY format. If no default is specified, Getdate uses the system date.

To make things easier, if the date is in the same month as the default, only the day of the month needs to be entered. If the default year is correct, only the month and day need to be entered. The date is assumed to be in the twentieth century unless the full year is entered as shown in Figure 1.

Getdate takes two parameters, both of which are optional. The first parameter is the default date, the second is a prompt. For example:

```
duedate:= @Getdate(@sumdate(invoice date,30),
                    "Date Payment is Due?")
```

Getdate could be entered into the formula of a spreadsheet cell or in the field heading of a database. Every time a spreadsheet containing a Getdate formula is recalculated, however, the dates must be reentered. Also, Getdate would need to be on the desktop or in the Library whenever the spreadsheet was loaded.

{Alt-D} is a macro that places a date value, obtained through a call to Getdate, directly into a spreadsheet cell. To use {Alt-D}, move or copy it into your Library Cabinet. Getdate must be on the desktop or in the Library. {Alt-D} requires that the spreadsheet be labeled. Highlight the cell that is to receive the date and press Alt-D.

```
; Getdate.FW2
; by Chuck Litzell
; November 1, 1987
;
; User-defined function to input a Framework II date.
;
; Format:
;
;       @Getdate(default, prompt)
;
; Both default and prompt are optional. If used, default
; must be a Framework II date. If it isn't specified, today's
; date is the default.
;
; User may enter:
;
;   1.  Nothing
;   2.  MM/DD/YY
;   3.  MM/DD/YYYY
;   4.  DD
;   5.  MM/DD
;
; If month or year are excluded, the default month or year is used.
;
;
;
@local(default, prompt, month, day, year, tdate, slash1, slash2),
```

```
@if(@or(@itemcount = 0, @iserr(@date1(@item1))),
   default := @today,
   default := @item1
),
@if(@itemcount = 2,
   prompt := @item2,
   prompt := "Enter Date"
),
tdate := @value(@integer(default)),
year  := @int(tdate / 10000),
month := @int(tdate / 100 - year * 100),
day   := tdate - year * 10000 - month * 100,
@if(year < 100,
    @list(
       year    := year + 1900,
       default := @date(year, month, day)
    )
),

@while(#TRUE,
   tdate := @inputline(prompt, @scientific(default), #NO, #YES, #NO),
   @eraseprompt,
   @if((slash1 := @scan("/", tdate)) = #N/A!,
      @if(@not(@iserr(day := @value(tdate))),
          @return(@date(year, month, day))
      ),
      @if((slash2 := @scan("/", @mid(tdate, slash1 + 1, 99))) = #N/A!,
         @list(
            @if(
               @and(
                  @not(@iserr(month := @value(@mid(tdate, 1, slash1 - 1)))),
                  @not(@iserr(day   := @value(@mid(tdate, slash1 + 1, 99)))),
               ),
               @return(@date(year, month, day))
            )
         )
      )
   ),
   @if(
      @and(
         @not(@iserr(month := @value(@mid(tdate, 1, slash1 - 1)))),
         @not(@iserr(day   := @value(@mid(tdate, slash1 + 1, slash2 - 1)))),
         @not(@iserr(
             @if((year := @value(@mid(tdate, slash1 + slash2 + 1, 99))) < 100,
                  year := year + 1900
             )
         )),
      ),
      @return(@date(year, month, day))
   ),
   @beep
)

; {Alt-D}
;
; Macro to set current selection to a Framework II date.
; Uses Getdate, which must be in the Library or on the desktop.
;
@local(func),
func := @setselection & " := @item1",
@func(@Getdate)
```

## #NAT

By Chris LoVasco

#NAT is an undocumented Framework II constant. It is the value returned by assigning (:= or @set) the @nationalize function to a reference. For example, the expression

```
Testframe := @nationalize(@yen)
```

returns #NAT to the frame Testframe. No value is returned by the @na**tionalize** function if an assignment function or operator is not used.

## @hlookup

The documentation for **@hlookup** on page 1-85 in *Advanced Topics Framework II* is misleading. The second parameter, which is the region to be searched, must be the entire region, including both the lookup row and all the rows containing data to be returned. Also, the example on page 1-85 is incorrect. Recalculated, it returns the error message "Offset value is greater than table size." This error occurs because the region referenced is a single row. It must contain at least four rows if an offset of three is to be used. The example should read

```
@hlookup(50, volbreak.f1:volbreak.j4, 3)
```

## Available Memory

The first time either the **@run** function is executed or **DOS Access** is used, Framework II loses some desktop/processing memory for the duration of the session. The actual amount of memory lost depends on whether extended memory is in use: Generally, more memory is lost if extended memory is in use. The memory loss is a result of Framework II's memory-management scheme.

## @sense

In *Advanced Topics Framework II* on Page 1-109, the documentation states that the **@sense** function returns a #TRUE or a #FALSE depending on the state of the toggle. It should also state that there are exceptions to that rule, as outlined on pages 1-110 and 1-111 and that numeric values may also be returned. The following table illustrates the parameters that return numeric values and the numeric values that they return.

| Parameter | Numeric Value Returned |
|---|---|
| #ATSSFD | 7 or 8 |
| #ATSSFP | 0, 1, or 2 |
| #ATSSFS | 1 or 2 |
| #ATSSMS | 1, 2, or 3 |
| #ATSSCR | 0 or 1 |
| #ATSSCD | 0 or 1 |

Figure 2

## IBM Screen Drivers

The screen drivers IBMSYS2G.SC and IBMSYS2M.SC do not support desktop color on the IBM Personal System/2 Model 30. Only the IBMSYS2T.SC driver supports a two-color desktop. However, while in **ZOOM (F9)** graphs are displayed in color for all three drivers.

## Automatic Recalc of the Library

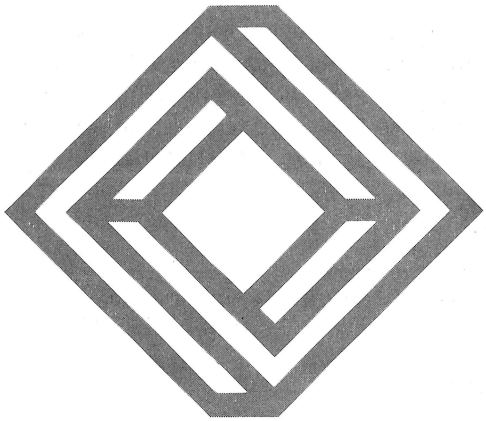
Framework II has a startup option that loads and recalculates a frame right after the Library is loaded. If you start Framework II by typing

```
FW framename/X
```

framename is loaded and recalculated. If you use the /X option without naming a frame, Framework will recalculate any formula on the border of your library.

This is convenient if you need a start up routine recalculated upon loading Framework II, such as setting keyfilters or executing an application you have developed.

If you want to put formulas in the border of your library, you must load Library.FW2 to the desktop separately, edit the formula, and then save it back to disk.



# ANOMALIES

## DEFINITION OF ANOMALY

We use the word "anomaly" to mean any one of the following:

- INCORRECT COMMAND (or function) IMPLEMENTATION. A command or function does not execute as specified.
- INCOMPLETE COMMAND (or function) IMPLEMENTATION. A command or function does not execute as specified in every use of the command or function.
- DATA OR LOGICAL BOUNDARY LIMITATION. A command or function cannot perform an operation at a specific data or logical boundary.

**NOTE**: Ashton-Tate reserves the right to withold any further information that may reveal proprietary or trade secret information.

dBASE III PLUS

## CONFIG.DB

Although not documented or implemented in dBASE III PLUS, including the line HISTORY = OFF in the CONFIG.DB file affects HISTORY. Specifically, if this is included prior to the line HISTORY=<n>, it has no effect. But, if it is included after HISTORY = <n> or if HISTORY = <n> is not included, this sets HISTORY OFF. Further, SET HISTORY ON has no further effect, HISTORY remains OFF until dBASE III PLUS is loaded again, though LIST or DISPLAY STATUS shows HISTORY is ON.

This line was a valid entry in CONFIG.DB in the Developer's Release. Users updating from this version should check their CONFIG.DB files for this entry. This occurs in both versions of dBASE III PLUS.

## COPY TO/COPY FILE

When you are using the COPY TO/COPY FILE commands and the target disk becomes full, answering "Abort" to the prompt "Disk is full when writing file: <Filename>.DBF Abort, Ignore, or Delete old files?" will leave the destination file open, and you will not be able to close it from dBASE III PLUS. Attempts to access the target file produce the error message "File is already open." [3]. Using the dBASE III PLUS DIR command indicates that the inaccessible file is "Not readable" and the size is zero bytes. To close the file, QUIT dBASE III PLUS.

When "Abort" is seleted, the prompt "** WARNING ** Data will probably be lost. Confirm? (Y/N)" is displayed. If the "Abort" is confirmed, the prompt "Press <-' to assent to the License Agreement and begin dBASE III PLUS" is displayed before control returns to the dot prompt.

Choosing "Ignore" in response to the prompt "Disk is full when writing file: <Filename>.DBF Abort, Ignore, or Delete old files?" redisplays the prompt. Choosing "Delete" produces a list of the files from

the target disk and displays the prompt "delete file: <Filename> ? (Y/N)" for each file. Answering "Yes" deletes the file and COPYing continues until finished or until the disk is full again. Answering "No" to the "delete file: <Filename> ? (Y/N)" prompt displays the next filename in the disk directory. Included in the list of files that may be deleted is the target file, but attempting to delete the target file produces the error message "Cannot erase a file which is open." This occurs in versions 1.0 and 1.1 of dBASE III PLUS.

## EDIT

When EDITing a numeric field, use of the **Del** key or **Ctrl-G** in the integer portion of the number may delete numbers from the decimal portion of the number. If the **Del** key or **Ctrl-G** is used to delete all the numbers from the integer of the number, the decimals remain intact. However, if only a portion of the integer is deleted, the decimal portion of the number may be replaced with zeroes. This occurs in every version of dBASE III PLUS.

## File Handle

When the error message "Too many files are open." [6] is encountered in a program and there is no ON ERROR, SUSPENDing the program and closing database files may leave a file handle open if the error message is encountered again and a database file is in USE. For example,

```
* Test.PRG
*
* ---Recursive calls to produce the error.
DO Test
* EOP: Test.PRG

* ---From the dot prompt, type:
* ---Disable ON ERROR
ON ERROR
DO Test
```

When the error prompt displays, choose SUSPEND. From the dot prompt, enter

```
RETURN          && Close one .PRG file.
RETURN          && Close another .PRG file.
USE Filename
DO Test
```

Test produces the error message "Too many files are open." [6]. At the "Cancel, Ignore, or Suspend? (C, I, or S)" prompt choose SUSPEND.

```
CLOSE ALL       && Or CLEAR ALL.
DO Test
```

The program again produces the error message prompt. Choose CANCEL. At this point one file handle is lost. The number of file handles lost is equal to the number of database files closed in the CLOSE ALL command. You must QUIT dBASE III PLUS to gain access to the lost file handle. This occurs in both versions of dBASE III PLUS.

## F1 Key

The BROWSE cursor-control menu says **F1** calls Help. Instead, **F1** toggles the cursor-control menu on and off in both versions of dBASE III PLUS.

## INSERT

INSERT with SET CARRY OFF adds an incorrect record if **PgDn** or **PgUp** (version 1.1 only) is pressed instead of **Ctrl-W**. If no changes are made to the record, pressing **PgDn** in version 1.0 appends information from the record following the INSERTed record instead of leaving it blank. In version 1.1, both **PgDn** and **PgUp** append the information. If any field is changed, the data are saved as expected.

## MOD()

The MOD() function returns incorrect results if the second argument evaluates to zero. It is not documented that the second argument of the MOD() function must not evaluate to zero. If the second argument of the MOD() function evaluates to zero, the value of the first argument is returned instead of an error message. For example,

```
? MOD(14,0)
    14
? MOD(-3,0)
     -3
mvar1 = 232
mvar2 = 0
? MOD(mvar1,mvar2)
   232
? mvar1/mvar2
*****
```

These should produce an overflow or an error message, (note the last example), because division by zero is an undefined concept. This error occurs for every version of dBASE III PLUS.

## MODIFY COMMAND/FILE

A file with an eight-character filename and no extension cannot be loaded with MODIFY COMMAND/FILE. Attempting to MODIFY such a file produces the error message "File is already open." [3]. To work around this, add an extension to the filename or shorten the filename and use two periods after the filename. A file can be created

with an eight-character filename and no extension, but it cannot be reopened once it is closed. The default filename created by MODIFY FILE has no extension. This occurs in versions 1.0 and 1.1 of dBASE III PLUS.

## SET CATALOG ON

With SET CATALOG ON in dBASE III PLUS version 1.1, pressing **Esc** while a file-opening command is executing changes the workarea to ten (the catalog workarea) and may eventually hang the system requiring a cold boot. When a file-opening command is executing, dBASE III PLUS checks to see whether the file is in the catalog. To do this, it temporarily selects workarea ten. If **Esc** is pressed while the catalog file is selected, the current workarea remains area ten once command execution is interrupted. At this point, memory has become corrupted. Subsequent commands may hang the system requiring a cold boot. For example,

```
SET CATALOG TO Test
*
* ---To ensure workarea 1.
SELECT 1
USE Test INDEX Test1, Test2,;
   Test3, Test4, Test5
```

Pressing **Esc** while USE is executing resets the current workarea to ten. In the example, five index files are opened to increase the time it takes dBASE III PLUS to open the files and improve the chances of producing the error. Issuing QUIT once the error has occurred may hang the machine.

In version 1.0, pressing **Esc** while the catalog is being checked produces the prompt "Enter title for the file <Filename>.dbf:" and adds the database and index files to the catalog; duplicating entries in the catalog that already exist. After leaving the prompt, the current workarea is reset to the original workarea. The system does not hang.

## SET COLOR TO

dBASE III PLUS does not reset the screen border color when the SET COLOR TO command contains the same border parameter as the previous SET COLOR TO command. For example,

```
*
* ---Set the border color to blue.
SET COLOR TO ,,B
*
* ---Clear the entire screen,
* ---including the border.
RUN CLS
*
* ---The border stays black.
SET COLOR TO ,,B
```

To reset the border to blue, first set it to another color and then to blue.

```
SET COLOR TO ,,R   && Set border to red.
SET COLOR TO ,,B   && Set border to blue.
```

In dBASE III PLUS versions x100 and x102, SET COLOR TO ,,<Border> has no effect on EGA monitors. The border color is set to black when dBASE III PLUS is loaded. The border option is available in the full-screen SET command, but it has no effect.

On an IBM PS/2 with a VGA (Video Graphics Array) display, SET COLOR TO ,,<Border> and COLOR=,,<Border> in CONFIG.DB work correctly. However, the border option is not available in full-screen SET. This is true for every version of dBASE III PLUS.

## SET FORMAT TO

APPEND corrupts database files when an active format file contains an alias name of "M" and memory variables with the same name as field names exist. When the format file includes M-> aliases, the memory variables, not the fields, are edited.

For example, to re-create this anomaly, declare an alias of "M." This causes CREATE SCREEN to generate a format file with the M-> alias for each field. Then declare memory variables to correspond with each field in the database file. The fields and memory variable must have identical names.

```
USE Test ALIAS M
*
* ---Load fields and save the screen.
CREATE SCREEN Test
USE Test
SET FORMAT TO Test
SET STATUS ON
APPEND
```

As records are APPENDed the status bar reveals that the record counter is incremented. After the changes are saved, LIST STRUCTURE reveals that the record count matches the status bar.

When APPEND is used with the format file active, dBASE III PLUS increases the database file record count but no data are added. Issuing a GO to the record that was the EOF() prior to APPEND produces the error message "Record is out of range." [5]. LIST does not go past the former EOF(). But, GO to a record past the former EOF() executes

without error. DISPLAY reveals garbage or blank data in these records. This occurs in both versions of dBASE III PLUS.

## SET PROCEDURE TO

In dBASE III PLUS, procedure files cannot call program files that then call other procedure files. The error message "File is already open." [3] occurs when a program file called from a procedure file contains SET PROCEDURE TO. For example,

```
*Test1.PRG
SET PROCEDURE TO Test_pr
DO Proc_1
* EOP:Test.PRG

*Test_pr.PRG
PROCEDURE Proc_1
DO Test2
* EOP:Proc_1
* EOP:Test_pr.PRG

*Test2.PRG
*
* ---Produces the error.
SET PROCEDURE TO Test_it
* EOP:Test2.PRG
```

The error message occurs when the SET PROCEDURE command in Test2.PRG is executed. This is true for every version of dBASE III PLUS.

## USE

Database files with one million (1,000,000) or more records hang the system when any of the following commands are executed: APPEND, APPEND BLANK, BROWSE, EDIT, CHANGE, or READ. APPEND appears to execute correctly until the command is terminated, then either the system hangs or the error message "Divide overflow" is produced and control returns to DOS. If dBASE III PLUS is restarted after the "Divide overflow" error message, the system eventually hangs. If "Divide overflow" is encountered, reboot the computer.

Depending on the amount of data entered before APPEND is exited, the file record count may or may not be increased. If the buffers are flushed to disk before the system hangs, the file structure is updated. However, the added records may or may not contain data.

All other file access commands operate correctly with one million or more records. This occurs in both versions 1.0 and 1.1 of dBASE III PLUS.

FRAMEWORK II

## @while

In Framework II, when a function that returns a string value (such as **@prompt**, **@integer**, **@decimal**, and **@textselection**) is executed from within a formula, the amount of available memory decreases until the formula terminates. Available memory is decreased because the string values that are returned by these functions are stored in a "string" buffer in memory. The string buffer can grow, but cannot shrink until the formula executing the functions terminates. Under normal conditions, this is not a problem. However, if the function is executed repeatedly from within an **@while** loop, it is possible for the string buffer to exceed its 64K maximum. When this happens, one of the following errors may occur:

- The error message "Too many string references" is displayed.
  This is the normal error message. It indicates that expanding the string buffer any more will exceed the limit.
- The error message "Internal Error detected, Restart DOS and Framework II." is displayed. When this error message is displayed, the formula is terminated, and the user should restart Framework II or the system may hang.
- The error message "Out of memory" is displayed. When this error is displayed, the formula is terminated, and the user should restart Framework II or the system may hang.
- The system may hang.

To demonstrate this anomaly, create an empty word frame and enter the formula

```
@while(#TRUE,
    @prompt(@business(@memavail))
)
```

on the frame border. During recalculation, the prompt displays the decreasing amount of memory. Please note that the **Max times @while loop executes** option, in the Framework II Setup program, may have to be increased to avoid encountering the "Exceeded number of @while iterations allowed" error message before one of the above errors occurs.

# ON-LINE SERVICES

By Steve Silverwood

Do you have a modem? Do you subscribe to CompuServe or The Source? If so, Ashton-Tate offers technical support on all of these electronic information services. You can quickly access the kind of quality technical information that you have come to expect from ***TechNotes***, and you can do it through your computer system. To take advantage of the progressive support method, you must subscribe to one of the services listed below. There are no additional charges for using our on-line support services, apart from the normal connect-time fees for each service. You can get specific information by calling one of the following numbers:

| Subscriber Service Information | | Command to find Ashton-Tate |
|---|---|---|
| CompuServe | (800) 848-8199 | GO ASHTON |
| The Source | (800) 336-3366 | ATSIG |

To give you an idea of the support that is happening on-line, the following are regular excerpts from the Ashton-Tate Forum on CompuServe. This monthly material comes from messages our Forum users have posted on the bulletin board. All message headers have been removed to protect the identities of our users. Every other message within a single conversation topic has been italicized to enhance readability.

## About On-Line Support

Ashton-Tate supplies product support through CompuServe and The Source to supplement telephone support. If you need technical assistance and the problem is not an emergency or occurs after the Support Center is closed, you can pose your question to the members of either of our two forums. You'll be answered not only by Ashton-Tate technicians but also by other forum members.

On-line support gives you immediate access to programs and utilities, which you can download to your computer from the forum at any time. Programs available include file-recovery utilities, source code formatters, and even complete applications. The authors of these programs are often available on the forum to answer questions or accept feedback about how to improve the capabilities of their programs.

The message base on each service gives you access to all the current questions and answers. Another member's question might help you find solutions. On our CompuServe forum, many of the members are professional software developers who use dBASE III PLUS and Framework II to create complete applications for their clients and who have an excellent working knowledge of techniques that often can help you over a hurdle or two.

## Recent Discussions

Some of the recent topics in ASHFORUM on CompuServe include:

- The recently formed dBASE Standards Committee
- Walter Kennamer's dBASE source code documenting program, SNAP!, which is available for downloading from the forum
- Add-on programs for use with dBASE
- Shareware and public-domain programs for dBASE applications

The discussions are lively, spirited, and sometimes even heated, with many of opinions being exchanged. Between the 9,000-plus ASHFORUM

members, at least *someone* has used (and has opinions about) just about every dBASE and Framework utility that's ever existed, and some of the companies producing those utilities (Wallsoft, for example) visit the forum regularly.

## New Services and Features

Ashton-Tate's Customer Service department is now represented on CompuServe by Shelly Landry. Shelly can answer questions about updating or upgrading existing Ashton-Tate products, to get help in ordering replacement disks, and other services provided by that department. A section of the forum message base has been set aside just for this service, so you can drop Shelly a note on the forum.

Support is now available on both The Source and CompuServe for RapidFile, and on CompuServe for the MASTER GRAPHICS line of products. And Ashton-Tate's newest product, dBASE Mac, is supported on both The Source and CompuServe.

## Automated Access

There are some software programs that can make accessing the on-line forums more efficient. These programs, tailor-made for either The Source or CompuServe, allow you to prespecify the names of the forums you want to visit, automatically check the messages (and in some cases the files available for downloading), and then capture this information to your computer's disk drive. This means that you can spend your time doing other work while the computer checks your mail, thereby saving connect time (and money).

For CompuServe users, one of the most popular programs of this type is ZAPCIS, written by Howard Benner of the IBMCOM forum and released as a shareware product. ("Shareware" describes software that you can download and use immediately; if you find it of value you, then send a check to the author.) The program and its documentation can both be downloaded from the IBMCOM or CONSULT forums on CompuServe.

ZAPCIS allows you to set up as many as 20 forums for regular access. With ZAPCIS, you can read messages off-line, select threads to be read after a "quick-scan," and compose and edit replies or new messages. The program can also search the data libraries for keywords or filenames. A public domain program that does a similar job to ZAPCIS is AutoSIG (also known as ATO). This program, also available on IBMCOM, provides the same message functions as ZAPCIS, but does not provide automated access to the data libraries unless you construct special scripts. Both programs offer a terminal mode for interactive CompuServe access (such as getting stock quotes, checking news articles, and uploading files) and both use the CompuServe "B" protocol for file transfers. Also, both are specific to the IBM PC.

A Macintosh program, called Navigator, is available from the Mac Users (MACUS) forum on CompuServe. More information on the package is available from that forum.

For users of The Source, a commercial program has recently been made available called ReSource.

## Data Library

Rather than present message threads in this issue, we've decided to publish a short listing of the files available in the dBASE III PLUS data library on CompuServe (DL2). This listing is current as of September 1, 1987. To find the new files, you can BRO /AGE:60 (where 60 stands for 60-days old).

**Note:** Items listed with "(TN, 5/85)" indicate that the file concerned is a ***TechNotes*** file, from the issue dated as shown.

## Files from the CompuServe Forum Data Library

| File | Size (K) | Description |
|---|---|---|
| 123DB3.ART | 3 | Data transfer, dBASE III to 1-2-3 (TN, 5/85) |
| AMORTC.TXT | 10 | Amortization using Ctools (TN, 12/86) |
| AMT.PRG | 3 | Draw on financial investment |
| ANOMLY.TXT | 13 | dBASE III PLUS known anomalies as of 7/86 |
| ANYE.DES | 3 | Description of pcAnywhere |
| APPEND.ARC | 10 | Thread on the APPEND FROM command |
| ARCBAC.ARC | 9 | Backup subdirectories within dBASE via ARC |
| ASM.ARC | 13 | Assembly-language interface in dBASE |
| ATMEET.ARC | 24 | Notes on 1986 Ashton-Tate Developers' Conference |

```
BACKUP.ART       4   Auto file backup in dBASE III (TN, 5/85)
BAR.ARC          1   dBASE III and III PLUS status bar program
BARGRA.ARC       2   Generates a bar graph in dBASE
BETDAT.PRG       3   calculates # of months/days between days
BILLIN.ARC      32   Time billing for services
BROWSE.PRG       1   Simulate the BROWSE command in a program
CAPRES.ART       7   "Capturing & Restoring SET Environments" (TN, 7/87)
CBIN.ARC        20   BIN routines for dBASE III PLUS
CBIN3.ARC       18   Corrections to CBIN.ARC
CHARS.ARC        5   Displays all 256 chars ;& hex/dec values
CHECKP.ARC       2   Checks printer status
CHMOD.ARC       23   Changes file attributes
CLEAN2.ARC      22   dBASE program file filter (strip blanks, comments, etc.)
CLOSES.ARC       1   Close spool file on 3COM 3+ network
CLRTSR.ARC       5   Clear TSR programs from memory
COLOR.LBR        1   Display all color combinations available (dBASE III, DR)
COLOR.LQR        3   Custom installation of mono/color screen attributes
COLORS.ARC       4   Set screen colors with dBASE III PLUS
COLORS.ARC       2   Thread on dBASE color commands
COMMAS.EXE       8   C program to write comma-delimited file w/o quotes
COUNT.ARC       14   Line counter written in Turbo Pascal
CPDINT.PRG       1   Calc compound interest amt based on starting principal
CURONO.ARC       1   Russ Freeland's CURONOFF.ARC program
CURSOF.BIN       1   BIN file to toggle cursor off
CURSON.BIN       1   BIN file to toggle cursor on
D3FIX.DOC       11   Documentation for DB3FIXPO.EXE (here as D3FIX.EXE)
D3FIX.EXE       26   dBASE III file recovery utility by Pete Olympia
DATASC.ARC      19   Scroll through data file via PRG
DATTIM.TXT      22   Assorted date and time routines (TN, 11/86)
DB3CAR.ARC       2   dBASE III menu card -- sample program
DB3PRT.ART      21   Special printing capabilities (TN, 6/85)
DB3SPS.COM       2   dBASE-to-SPSS translate utility
DBASE3.ARC      95   dBASE III information (from Support Library)
DBASEF.DOC       6   Documentation for DBASEFIX.EXE (here as DBASEF.EXE)
DBASEF.EXE      46   Formats PRG files
DBDIR3.ARC       9   Simulates dBASE's DIR command from within DOS
DBFCRE.ARC       8   Converts .DBFs into .PRGs to re-create the database
DBSTDS.ARC      49   STANDARD.THD -- dBASE Standards Committee thread
DBUGDP.ART      20   "Using DEBUG with dBASE" (TN, 7/87)
DBWIND.ARC      14   Window program
DBXREF.COM      17   Cross-references dBASE procedure files
DCNT09.LQR      32   PRG file cleanup utility (strip blanks, comments, etc.)
DCRUNC.ARC      19   PRG file cleanup utility (strip blanks, comments, etc.)
DELDUP.PRG       2   Delete or print duplicate records
DIAL.PRG         1   Phone dialer from within dBASE
DICT.ARC        16   Data dictionary
DINDEN.COM      14   PRG formatter/indenter
DIREXI.ARC       2   BIN routines for dBASE III PLUS
DISKSP.ARC       2   Check available disk space in any drive
DOLLAR.PRG       1   PLUS procedure for formatting dollar amounts
DOSMOD.ARC       4   Allows toggling of DTR within dBASE
DSCAR2.ARC      64   dBASE program source code formatter
DSPNDX.DOC       3   Documentation for DSPNDX.EXE
DSPNDX.EXE       2   Display key expression for index file from DOS
DSTRIP.COM       1   PRG file cleanup utility (strip blanks, comments, etc.)
EDI.ARC        150   Edit, display, search records from DOS
ENCRYP.ARC       1   Encrypt/decrypt strings
EXPORT.ARC       3   Thread on exporting from dBASE
FFL.ARC         14   Fast file lister in Turbo Pascal
FILNAM.ARC      10   Thread on validating dBASE filenames
```

```
FIXES.TXT       12   Fixed anomalies in dBASE III PLUS
FLDDTO.ART       4   Field ditto during data entry (TN, 3/87)
FMTOUT.ART       4   "Formatting Output" (TN, 8/87)
FRM2PR.ARC      17   Converts FRM files to PRG files
FRMTMP.TXT       6   Report form templates in dBASE III PLUS (TN 12/86)
FSIZE.ARC        3   Find size of a file or group of files
GENIMP.ARC      10   "Generic Import/Export of Database Files, p1" (TN, 9/87)
GENIMP.ART      15   "Generic Import/Export of Database Files, p1" (TN, 9/87)
GENSAM.ARC      18   Genifer sample generated code
GETDRI.ARC       1   Sample BIN routine to get the current drive
HANDLE.ARC       3   File Handle Status (TN, 12/86)
INDEX.ART        6   dBASE III date fields in NDX expr. (TN, 8/85)
INKEY.ARC        4   Replace INKEY() with INKEY.BIN (TN, 4/87)
INKEY.ART        7   Replace INKEY() with INKEY.BIN (text) (TN, 4/87)
ISDISK.ARC       2   Check disk drive status from within dBASE
ISMEMO.ARC       4   Checks for existence of memo field for a given record
JMEMOS.ART      32   "Joining Databases Incl. Memo Fields" (TN, 8/87)
LABELS.ART      12   Overcoming limits of LABEL FORM (TN, 8/85)
LB28.ARC         6   LiteBar routine by Russ Freeland
LBFX41.ARC       3   Fix for LITEBAR version 4.1
LINEAT.COM      15   Source code formatter
LITEBA.ARC      26   Version 4.1 of Russ Freeland's Litebar programs
LITEBA.ARC      49   V 4.2 of Russ Freeland's LITEBAR program
LJUST.ART        2   Left-justify numerics (TN, 7/85)
LPRINT.COM      13   Source code printer & X-referencer
LPTX.ARC        19   Send @..SAY output to disk
LSTNUM.ART       2   "Select Fields to List by Numbers" (TN, 9/87)
LSTPIK.ART       8   "ListNPik" (TN, 9/87)
LSTSTR.COM      14   Lists .DBF structure in DOS
MAKEDB.ARC      13   Creates .DBF files in DOS
MAKEME.ARC      38   Creates .MEM files in DOS
MAKEUS.ARC       2   Demo of ENCODE and DECODE in ENCRYPT.ARC
MALLOC.ARC       7   Thread on dBASE III PLUS memory allocations
MDBFDE.ARC      36   Demo for v1.11 of MAKEDBF
MEMBER.ARC      32   Membership program w/custom user interface
MEMOS.ART       15   "Memo Fields Revisited" (TN, 5/87)
MEMPAK.ART       6   Compressing memo file contents (TN, 9/86)
MEMTAB.ASM       6   dBASE BIN routine
MEMTAB.DOC       5   Documentation for MEMTAB.ASM
MENU.ARC        22   Saywhat?! demo menu system
MENU.PRG         3   Lite-bar menu using INKEY()
MENUBA.ARC       2   Lightbar menus
MFRMT.ART        9   Multiple format files (TN, 9/85)
MIS.ARC         33   Processing dept. management application
MKDBF3.ARC      12   Creates .DBF files
MM-DP.TXT       11   MultiMate Advantage intfc to dBASE (TN 11/86)
MODEMS.ARC       5   Thread on modem access from within dBASE
MOSS.ARC         7   User-friendly interface example
MOUSE2.ARC       5   Microsoft Mouse access from dBASE
MPTLOF.TXT      19   Info on multiple-of-four problems
MRGAPP.ART      18   Mailmerge applications (TN, 9/86)
NO2WOR.PRG       1   Convert numbers to words
NOVINS.ART      15   "Installation on the Novell Advanced NetWare LAN" (TN, 9/
NUHED.ARC       14   .DBF file header repair utility
NUHED3.ART       2   .DBF file header repair utility (TN, 2/85)
NUMLOC.ARC       1   BIN file to toggle NumLock mode
NUMSOR           1   Sorting numbers in character fields
NUMWRD.ARC       8   Converting numbers to words (TN, 3/87)
NUMWRD.ART      18   Converting numbers to words (text) (TN, 3/87)
NUMWRD.FIX       1   Fix for the NUM2WORD.BIN file from TN 3/87
ONERRO.ARC       5   Experimental on-error routine for dBASE III PLUS
```

```
PACKFI.ARC       8   Controlled PACK/REINDEX routine
PACKIT.ARC      30   Reduce packing time by 50%
PAGES.ARC       13   Page back/fwd through screen displays
PASSWD.LBR       4   Password routine for dBASE III
PCNET.ART       32   "Fine-Tuning the IBM PC Network" (TN, 8/87)
PRCHEC.ARC       3   Check printer status with CALL to BIN file
PRFILE.ARC       2   Send @..SAY output to disk
PROPER.ARC       4   Convert memvar string to proper name
PRSTAT.ARC       7   Thread on dBASE printer status checking
PRTCTL.TXT      19   Control codes to printer (TN, 12/86)
PRTDRV.ARC       4   Create "printer drivers" for dBASE
PRTDRV.ART      15   "Creating dBASE Printer Drivers" (TN, 5/87)
PRTSWP.ART       4   Swap between LPT1 & LPT2 (dBASE III, D/R)
PRTSWP.COM       1   Swap between LPT1 & LPT2 (dBASE III, D/R)
PTOOLS.ARC      28   Turbo Pascal utilities for dBASE III PLUS
REPLAC.ARC       5   "Replacing One Character With Another" (TN, 7/87)
REPLAC.ART      11   "Replacing One Character With Another" (TN, 7/87)
RETDRV.ARC       4   "Retrieving Drives & Changing Directories" (TN, 9/87)
RETDRV.ART       7   "Retrieving Drives & Changing Directories" (TN, 9/87)
RPRN.ARC         3   Send @..SAY output to disk
RPTFOR.ARC      26   Articles on REPORT FORM command by Pat Adams
S-JOIN.PRG       2   Simulate the JOIN command via PRG file
SAVER1.ARC       2   Save/restore screens to/from memory
SAVERE.ARC      10   Save and restore screens to file (TN, 1/87)
SAVRES.ARC      20   "Save & Restore Partial Screens" (TN, 6/87)
SAVRES.ART      40   "Save & Restore Partial Screens" (TN, 6/87)
SAY2FI.ARC       6   Send @..SAY output to disk (TN, 6/87)
SAY2FI.ARC      10   "Sending @..SAYs to Text File" (TN, 5/87)
SAY2FI.ART      15   "Sending @..SAYs to Text File" (TN, 5/87)
SCORE.PRG        2   Statistical analysis
SCROL2.ARC       8   Scroll all or part of screen
SETCSR.ARC       1   Turn cursor on or off
SETENV.ART       7   "Generating SET Environment Programs" (TN, 8/87)
SETPRI.BIN       1   Printer control code utility
SETSET.PRG       4   User-definable monitor colors within program
SHARE.ART       14   SDF/DELI file structures (TN, 6/85)
SHOWRC.PRG       2   Scroll thru records, select desired one
SL.COM          42   Source debugger "similar to dFLOW"
SL.DOC           3   Documentation for SL.COM
SNKCOL.ART      15   "Snaked Column Reporting" (TN, 6/87)
STARLN.ART      28   dBASE III PLUS & AT&T Starlan (TN, 4/87)
SWAPPA.LBR       1   Allows writing pages off-screen then displaying them
TABLAD.ARC       1   Returns memory address for memvar symbol table
TAPEBA.DOC       1   Documentation for TAPEBA.PRG
TAPEBA.PRG       8   Backup utility using Compaq tape drive
TESTIN.PRG       2   Correction to TESTINT.PRG in CBIN.ARC
TIME.ARC         6   [76320,302] Store time of day in date variables
TIME.ARC         6   [76146,371] Andy Schulman's info on dates in dBASE III +
TWENTY.ARC       3   Open up to 20 files in dBASE
UNQFIL.ARC       5   Find unique filename for temporary work files
USING.ART        2   "Using $ Operator with READKEY()" (TN, 9/87)
VERIFY.ARC       1   Reset DOS verify switch
VIDEOL.ARC      18   Videocassette library program
WDB.ARC          2   Superkey macro for dBASE
WINDOW.ARC      21   Saywhat?! windowing routines
WINDOW.DOC       1   Documentation for WINDOW.ARC
WNWOAN.ARC      22   Windows for dBASE III PLUS
WORDWR.ARC      11   Another wordwrap (TN, 2/87)
WPMERG.ARC      17   Convert to WordPerfect for mailmerge
XTAB.PRG         2   Cross-tabs in dBASE
XTRACT.PRG       2   Wildcard search through data in file
```

# LETTERS

dBASE III PLUS

## Printing Leading Zeros

Dear *TechNotes*: Sometimes I need to print out a number on a report with leading zeroes: I need dBASE to display "00077" instead of "77". If the number is stored in a character-type field, there is no problem, but how can I display leading zeros when the fields are numeric? — LO

*Dear LO: A simple way to solve your problem is to use the STR and SUBSTR functions. Suppose you need to display the field EMPNO. If the actual value of EMPNO is 77 and you need to display "00077", you can add 100,000 to EMPNO and use STR to convert it to a character string:*

```
STR(Empno + 100000, 6)
```

*This will give you the string "100077". You can then remove the first character by using the SUBSTR function:*

```
SUBSTR(STR(Empno + 100000, 6), 2)
```

*The result is "00077".*

## Printing Program Listings

Dear *TechNotes*: I am using dBASE II to write some programs. Although I can display and edit programs with MODIFY COMMAND, I have not yet found a way to print out my programs without QUITting dBASE II and sending a copy to the printer from the operating system. How can I print out my programs while still in dBASE II? — TF

*Dear TF: One solution is to set up a database called Text.DBF with only one character field, which is 80-characters long. Then, while you are running dBASE II, you can print out a copy of Myprog.PRG by using the following commands:*

```
USE Text
DELETE ALL
PACK
APPEND FROM Myprog.PRG SDF
SET PRINT ON
LIST OFF
```

*These commands, after deleting the previous contents of the database, load each line of the program into record of the single-field database, and then print them without the record numbers. You could expand on this by writing a program (or a report) to embellish the simple listing.*

## APPENDing to Text Files

Dear *TechNotes*: I am writing a program in dBASE III PLUS that generates a custom report which sends the output to an alternate file. I want the alternate file to contain the data of all the previous sessions up to the current one. The problem I am having is that every time I SET ALTERNATE TO a file, dBASE III PLUS overwrites the file with the new data. How can I avoid this ? — OA

*Dear OA: To create an alternate file that accumulates data from different runs of an application program, create a database file called Text.DBF with a character-type field that is 80-bytes long. Add the following command lines to your application program to APPEND the lines of your alternate file (Altfile.TXT) to the Text database after the program has written the data to the alternate file.*

```
IF FILE("Altfile.TXT")
   USE Text
   APPEND FROM Altfile.TXT SDF
ENDIF
SET ALTERNATE TO Altfile.TXT
SET ALTERNATE ON
* Processing goes here
```

*Each time the program is run, the previous command lines will APPEND the data into the Text database file. After you have accumulated all the data you need, an option in the program can create the alternate file by using the following commands:*

```
USE TEXT
COPY TO Altfile.TXT SDF
```

*This technique can also be used for debugging purposes to compare data from different runs of the system.*

*Another solution might be to use the operating system to concatenate files for you. Instead of writing your data directly to the final output file, write it to a temporary file. You can then*

```
RUN COPY Tempfile.TXT >> Final.TXT
```

*which will create Final.TXT if it doesn't exist or will append information from Tempfile.TXT to the end of Final.TXT if it does exist.*

## Assigning Function Keys

Dear ***TechNotes***: I am developing an application that I want to make as user friendly as possible. I am trying to tell the user to press one of the function keys to get into memo field input mode. Normally, they would press Ctrl-Home to do this. Can you provide any ideas? — MF

*Dear MF: SET FUNCTION n TO CHR(29), where n is the number of the function key desired, can be used to emulate the* ***Ctrl-Home*** *key combination. CHR(29) is the code generated by a standard IBM keyboard for the INKEY() function.*

## Exclusive OR

Dear ***TechNotes***: How can I implement an exclusive OR in dBASE III PLUS? —UJ

*Dear UJ: The XOR logical operator is not directly available in dBASE III PLUS, but its functions can be duplicated.*

*(x XOR y)*

*is equivalent to:*

*(x .AND. .NOT. y) .OR. (.NOT. x .AND. y)*

FRAMEWORK II

## Disk Availability

Dear ***TechNotes***: I recently purchased a lap-top computer that takes only 3.5-inch diskettes. I have a copy of Framework that I run on my IBM PC. Can I get a copy of Framework in 3.5-inch format? — TH

*Dear TH: Framework is not available in a 3.5-inch format. However, Framework II is available in either 5.25 inch or 3.5 inch Format. Users of Framework can upgrade to Framework II in either of these formats. For more information on upgrading charges and procedures, please contact Ashton-Tate Customer Service at 213/329-9989.*

## Record Tagging

Dear ***TechNotes***: Very often, after running a batch of mailmerge letters or a list of labels, I realize some of the names need to be modified or deleted. How can I print the labels or letters again without having to reprint the entire database? — ML

*Dear ML: To print labels for a set of corrected records, create an extra field in the database that will let you "tag" the records you want to print. When you modify a record, enter a tag into that field. Any number or letter can be used to tag the records, as long as the same character is used for all the records modified in that session, and there are no other records with the same tag.*

*After all the modifications have been entered, filter the database on that "tag field" (TAG="Y"), and print this new set of records. When finished, delete the tags from the records, by highlighting the tag column and deleting all the records (a macro can also be implemented). This will leave the database ready for another edit session. For a full set of macros to implement this concept, see "Record Tagging" in the Framework Tips in the September 1987 issue of* **TechNotes** *on page 53.*

## Framework II on the Wang PC

Dear ***TechNotes***: Is it true that you have a version of Framework II that will run on the WANG PC? — IT

*Dear IT: The following are excerpts from an Ashton-Tate news release dated August 24, 1987:*

*"Ashton-Tate and Wang Laboratories, Inc., today announced that Framework II is available for use with Wang's Professional Computer (PC) and Advanced Professional Computer (APC).*

*"The product is sold through Wang and its authorized dealers.*

*"This new version of Framework II enables Wang word processing users to easily exchange files between the two systems."*

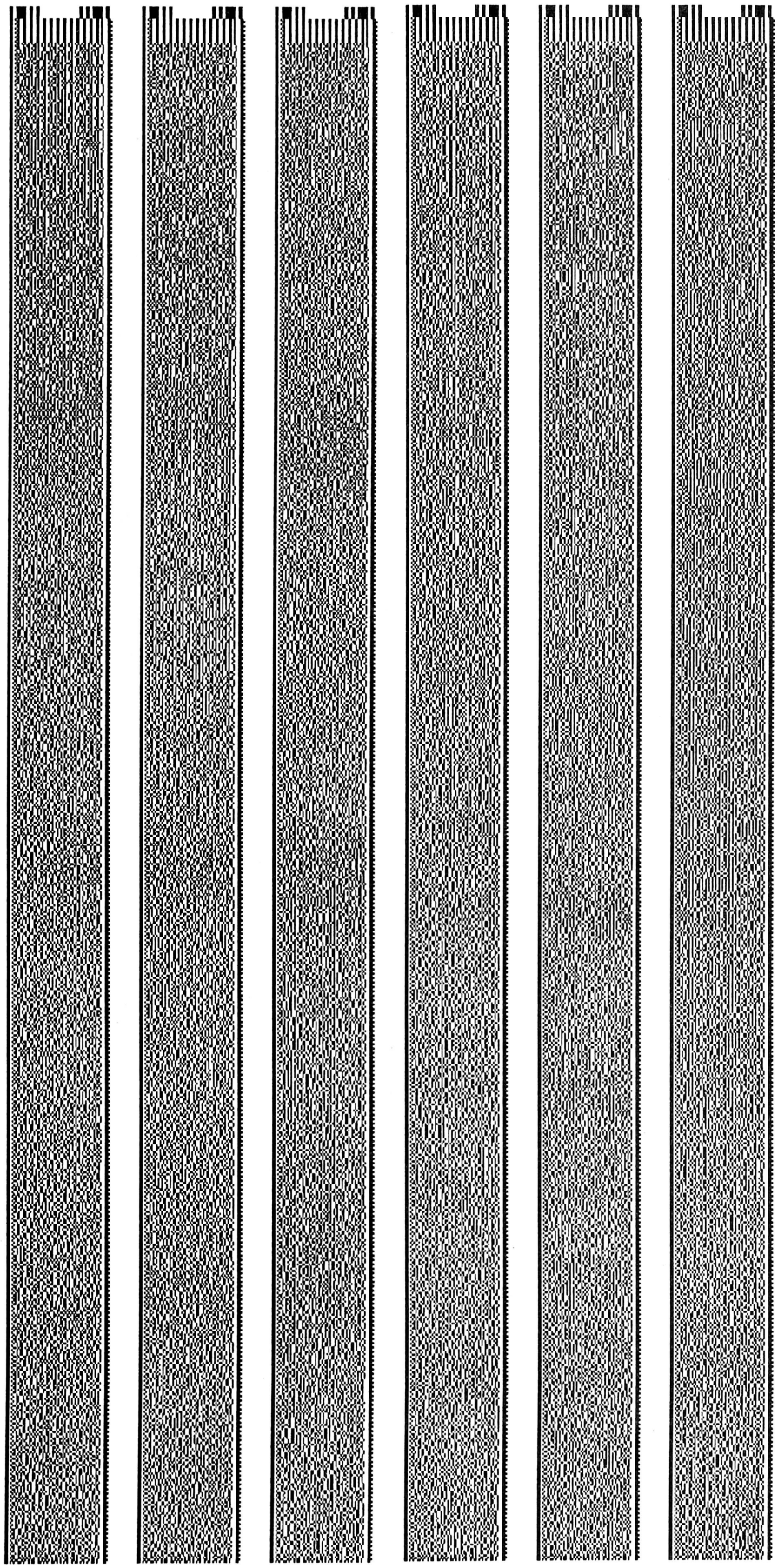
## @draw

*Create an empty word frame and place the following in its formula area:*

```
@local(drawform),
drawform := "@draw(4," &
       @integer(coords.B2) & "," &
       @integer(coords.C2) &",9," &
       @integer(coords.B3) & "," &
       @integer(coords.C3) &",9," &
       @integer(coords.B4) & "," &
       @integer(coords.C4) &",9," &
       @integer(coords.B2) & "," &
       @integer(coords.C2) & ")",
@execute(draw, drawform)
```

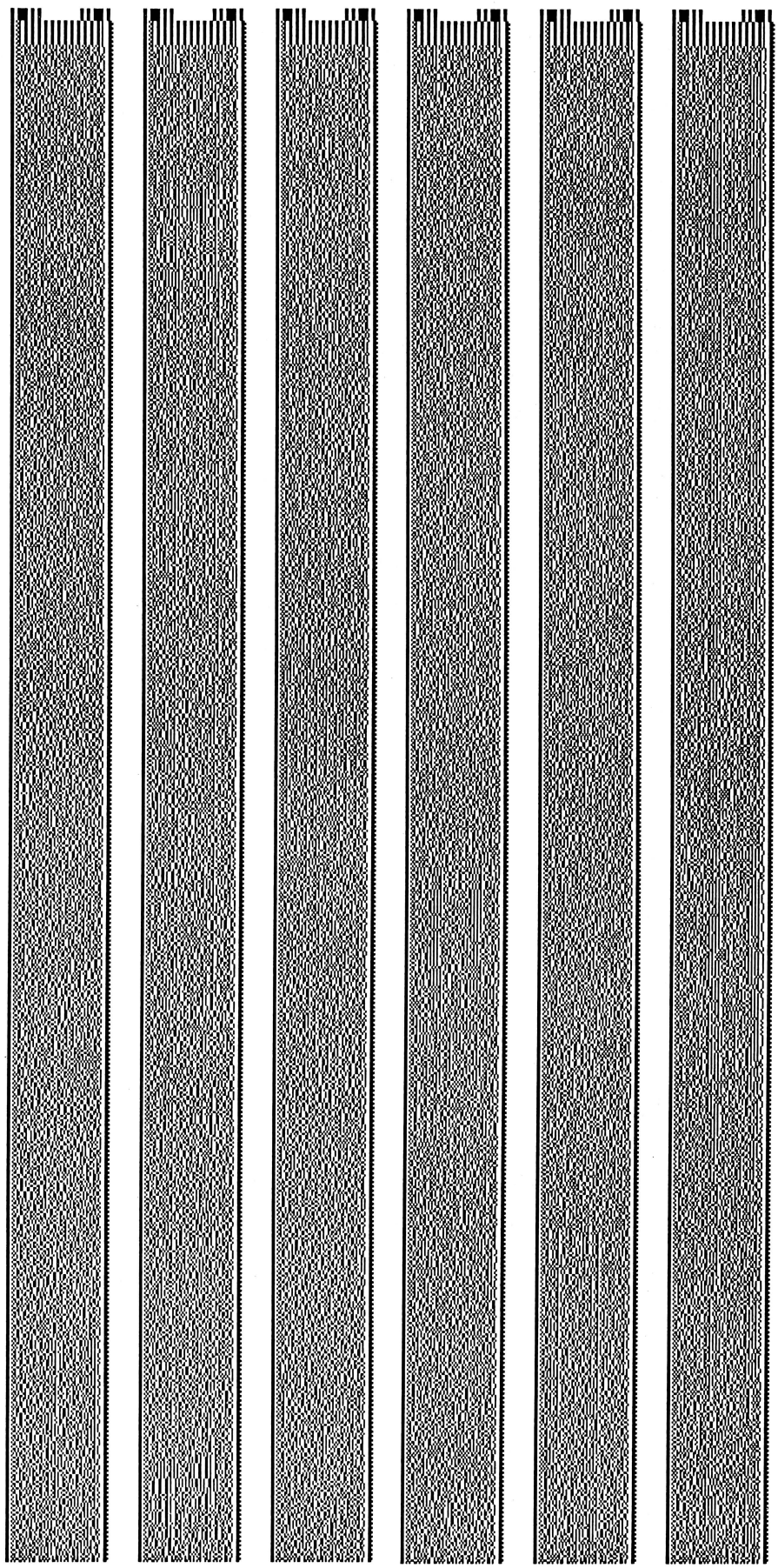
*Then create a second empty word frame called "draw." When calculating the program word frame, it will take the values on the spreadsheet, expand them into the* ***@draw*** *function and generate a triangle (in this particular case the coordinates form a triangle) in the draw frame.*

FILES:
ASCII.ARC
CNVTBASE.ARC
DSKSPACE.ARC
MAKESDF.ARC
MEMO.ARC



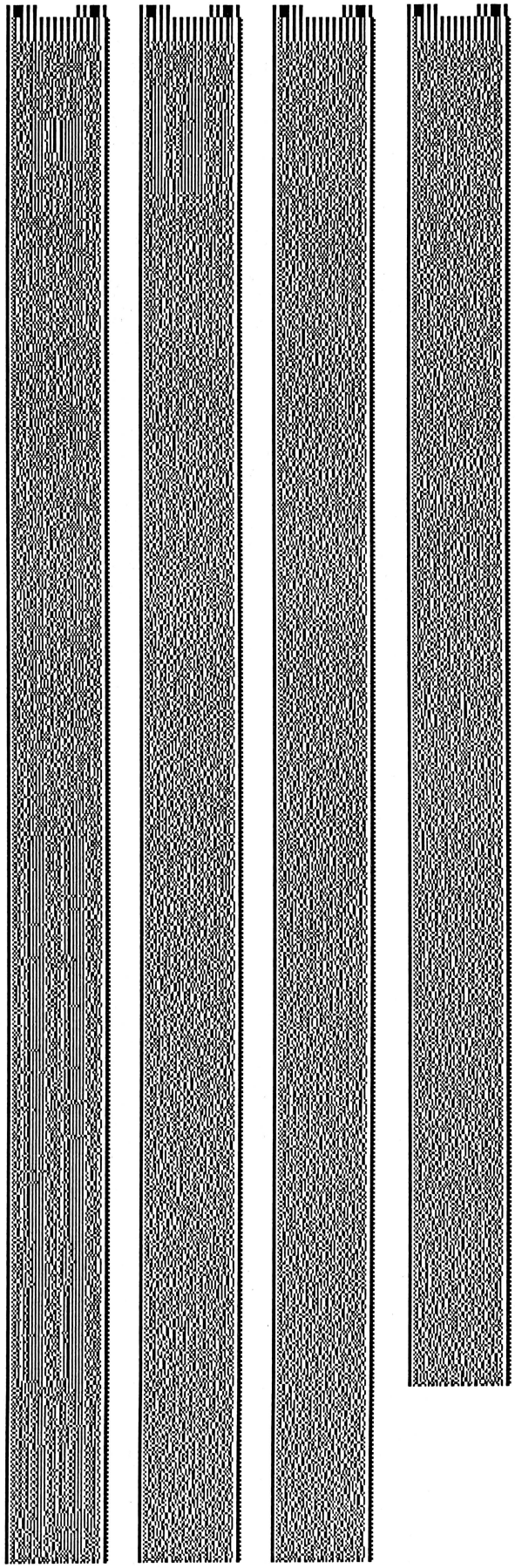
1 2 3 4 5 6

FILES:
ASCII.ARC
CNVTBASE.ARC
DSKSPACE.ARC
MAKESDF.ARC
MEMO.ARC



7 8 9 10 11 12

FILES:
ASCII.ARC
CNVTBASE.ARC
DSKSPACE.ARC
MAKESDF.ARC
MEMO.ARC



# CLASSIFIEDS

All descriptions of services and products have been provided by the advertiser. These products and services have not been tested by Ashton-Tate and their inclusion in this section should not be taken as an endorsement of them by Ashton-Tate.

## SOFTWARE/ APPLICATIONS

**SOFSOLUTIONS CO.**
**ATTENTION: Genifer is now Quicksilver/ Clipper Compatible.** This powerful utility automatically converts any Genifer generated program into Quicksilver/Clipper compilable format. Just enter your file name and the program does the rest. Saves hours of debugging time! For "Genifer to Quicksilver" order GeniQuick. For "Genifer to Clipper" order GeniClip. Unlimited tech support. $59.00

**To order by phone call (512) 735-0746 or send a check to SofSolutions Co., 208 Senisa, San Antonio, TX 78228**

## CONSULTING

**Ashton-Tate Support Technician**
Offering expertise in database management. Consulting, Programming, Training, etc... Haynes Enterprises, 818-886-6695

## USER GROUPS

Capital PC Users Group
Framework SIG
9420 Copenhaver Dr.
Potomac, MD 20854

Central Kentucky Computer Societey, Inc
Suite 100
Security Trust Building
Lexington, KY 40507

Chesapeake PC Users Group, Inc
P.O. Box 2365
Annapolis, MD 21404

dBASE SIG
St. Louis User's Group
for the IBM PC
P.O. Box 1345
St. Louis, MO 63043

dBASE Users Group
Picadilly Cafeteria
Centerville Rd and LBJ
Dallas, TX 73025

Greater South Bay PC Users Group
P.O. Box 1937
Redondo Beach, CA 90278-0937

Manasota IBM PC User Group
4500 Gulf of Mexico Dr. G301
Longboat Key, FL 33548

Space Coast PC Users Group
P.O. Box 396
Titusville, FL 32781-0396

Hawkeye PC User Group
P.O. Box 2966
Cedar Rapids, IA 52406-2966

University of South Florida
IBM PC Users Group
SVC 409
Tampa, FL 33620-7250

dBASE III Developers
Attn: Gil Orlik
2240 Buchanan, No. 5
San Francisco, CA 94115-9991

Philadelphia Area Computer
Society Framework SIG
2041 Harbour Drive
Palmyra, NJ 08065

Alaska Computer Society
6401 Downey Finch Dr.
Anchorage, AK 99516

San Diego Computer Society
IBM PC SIG
P.O. Box 87770
San Diego, CA 92138

Pemi-Baker Computer Users Group
RFD #2 Box 399
Plymouth, NH 03264

Richmond dBASE Users Group
4206 Seminary Avenue
Richmond, VA 23227

Tampa Database Users Group
119 Plantation Court, Suite D
Tampa, FL 33617-3731

The Boston Computer Society
One Center Plaza
Boston, MA 02108

Garden State Business Users Group
Lackawanna Plaza
Millburn, NJ 07041

Kansas City IBM PC Users Group
221 E. 3rd Street
Kansas City, MO 64113

UCLA PC Users Group
NPI 77-365, 760 Westwood Plaza
Los Angeles, CA 90024

Utah Blue Chips
P.O. Box 510811
Salt Lake City, UT 84151

NYPC dBASE SIG
40 Wall Street, Suite 2124
New York, NY 10005

Terre Haute PC User Group
100 South 22nd Street
Terre Haute, IN 47803

dBug SIG
Dallas, TX
(214) 495-8828


**MASTERMENU™** ONLY $37

**Instantly create/delete entire menu's**
**•Built-in menu generator •Passwords**
**•and more . . .**
(207) 439-5074
ATI, Lyndon Way, Kittery, ME, 03904


Picture City PC Programming Club
P.O. Box 20342
Rochester, NY 14602

Wiregrass Microcomputer Society
P.O. Box 1082
Enterprise, AL 36331

Hawaii PC User Group
P.O. Box 22967
Honolulu, HI 96822

Multi-Mate User Group
18314-48 Avenue West
Lynnwood, WA 98037

To receive an information sheet, call 213/538-7517

**Rates:** $1 per word. Minimum of 50 words. Submit copy and payment. We will reduce/ enlarge your art to fit available sizes.

**Payment:** Full payment must accompany each order. Mail copy and payment to: *TechNotes* Classifieds Attn: Brenda Johnson-Grau, 20101 Hamilton Avenue, Torrance, CA 90502-1319

**Agency Commission:** No agency commission on regular or display classifieds.

**Closing Date:** Ads must be received 15th of the month two months prior to cover date of magazine (that is, November 15th is the deadline for the January issue).

**Frequency discount:** 10% discount given for four consecutive insertions

**Acceptance:** Publisher reserves the right to accept or reject advertisements for Classifieds at their discretion.

**Tear Sheets:** A tear sheet showing classified ad as printed is sent without charge to each classified advertiser.

**Phone orders:** Not accepted.

**User Groups:** Ads are free. Name and address only. Limited to available space.


300 Personals

**STRESS ON THE JOB?**

Overworked? Burnout?
Depressed? Backache?

**Cheer up! We prescribe Genifer for dBASE programming without tears.**

**Genifer's the easy to use application generator that creates perfect applications in minutes.**

**Not only can it save you time, cranking out code. Genifer allows you to sit down with a user, listen to his or her requirements, and bring up a high-quality prototype in under an hour!**

**Call today for your free demo disk.**

**Bytel Corporation, 1029 Solano Ave. Berkeley, CA 94706. (415) 527-1157**
**1-800-631-2229 (CA 1-800-541-3366)**

## SUBSCRIBE TODAY TO TechNotes AND SAVE 30% OFF THE COVER PRICE !

Get Ashton-Tate's hottest source of technical know-how! Each monthly issue of ***TechNotes*** provides you with the most current, accurate technical information available from the technicians at our Software Support Center on all dBASE, Framework, and RapidFile products. Return this card today! Or subscribe by phone!

Call our toll-free numbers to subscribers: **(800) 331-4164** (in California) or **(800) 828-2514** (residents of other states). In Canada, call **(619) 487-3212**. Cover price is $6.

☐ 1 YEAR for $49.97 Save $22.03 ! ☐ 2 YEARS for $89.97 Save $54.03 ! ☐ 3 YEARS for $124.97 Save $91.03 !

Add $15 for Canada and Mexico. Add $25 for overseas.

☐ Bill me ☐ Check enclosed (Make payable to Ashton-Tate in U.S. dollars drawn on a U.S. bank)

☐ Visa: ☐ MasterCard:

Expiration date: Signature:

☐ New subscription ☐ Renewal

Name

Address

City State/Province Zip Code

Country Phone Number

Which product do you use? ☐ dBASE II (D2) ☐ dBASE III (D3) ☐ dBASE III PLUS (DP) ☐ Framework (FW) ☐ RapidFile (RF) Please allow 30 to 60 days for delivery of your first issue.

1000

## READER SURVEY

We would like to make sure that ***TechNotes*** continues to serve the needs of our readership. Please take the time to complete this survey.

Which Ashton-Tate product do you use?

☐ dBASE II ☐ dBASE III/dBASE III PLUS ☐ dBASE III PLUS Lan Pack
☐ Framework II ☐ RapidFile

How long have you been using these product(s)? _____ years _____ months

How long have you been a subscriber to ***TechNotes***? _____

Would you be interested in subscribing to ***TechNotes*** on disk? ☐ Yes ☐ No

Do you download the programs in ***TechNotes*** from the Ashton-Tate SIG on CompuServe or The Source? ☐ Yes ☐ No

Please rate these articles on a scale from 1 to 5 for accuracy, usefulness,and relevancy:

| | Accuracy | Usefulness | Relevancy |
|---|---|---|---|
| Hex to Decimal (and Binary Too) | ☐ | ☐ | ☐ |
| Memo to Character Conversion | ☐ | ☐ | ☐ |
| LAN Index Integrity | ☐ | ☐ | ☐ |
| Pop-up ASCII Table | ☐ | ☐ | ☐ |
| Finding Free Disk Space | ☐ | ☐ | ☐ |

## IF YOU'RE MOVING

Don't forget to tell us so that we can be sure your subscription to **TechNotes** and/or the Ashton-Tate Quarterly is uninterrupted. To process a change of address, we must have your previous mailing address (including the zip code) and your new address.

My new address is:

Name/Department

Company Name/Add'l Destination Information

Mailing Address

City/State Zip Country

Please affix your current mailing label(s) here. (If the label is not available, be sure to include your old address.)

Affix your current Ashton-Tate Quarterly mailing label here

Affix your current TechNotes mailing label here

To change your address on the Ashton-Tate Product Mailing List, necessary to receive upgrade and other important information about your Ashton-Tate product(s), you must send a separate change of address to: Ashton-Tate Customer Service, 20101 Hamilton Ave., Torrance, CA 90502-1319.

# CLAS

All descriptions o
have been provid
These products a
tested by Ashton-
in this section sh
endorsement of t

## SOFTWAI APPLICA

**SOFSOLUTION**
**ATTENTION: G**
**Clipper Compati**
automatically cor
program into Qu
able format. Just
the program doe
debugging time!
ver" order GeniQ
Clipper" order G
support. $59.00
**To order by ph**
**send a check to S**
**San Antonio, TX**

## CONSULT

**Ashton-Tate Supp**
Offering expertis
Consulting, Prog
Haynes Enterpris

## USER GR

Capital PC Users
Framework SIG
9420 Copenhave
Potomac, MD 20

Central Kentucky
Suite 100
Security Trust Bu
Lexington, KY 40

Chesapeake PC U
P.O. Box 2365
Annapolis, MD 2

dBASE SIG
St. Louis User's G
for the IBM PC
P.O. Box 1345
St. Louis, MO 63

dBASE Users Gr
Picadilly Cafeteri
Centerville Rd ar
Dallas, TX 7302

Greater South Ba
P.O. Box 1937
Redondo Beach,

Manasota IBM P
4500 Gulf of Me
Longboat Key, Fl

Space Coast PC U
P.O. Box 396
Titusville, FL 32

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**BUSINESS REPLY MAIL**
FIRST CLASS PERMIT NO. 66 TORRANCE,CA

POSTAGE WILL BE PAID BY ADDRESSEE

TechNotes

**Subscription Service**
P.O. Box 3729
Escondido, CA 92025-0929

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**BUSINESS REPLY MAIL**
FIRST CLASS PERMIT NO. 66 TORRANCE,CA

POSTAGE WILL BE PAID BY ADDRESSEE

TechNotes

**Reader Survey**
Ashton-Tate Software Support Center
20101 Hamilton Avenue
Torrance, CA 90502-1319

Attention: Brenda Johnson-Grau

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**BUSINESS REPLY MAIL**
FIRST CLASS PERMIT NO. 66 TORRANCE,CA

POSTAGE WILL BE PAID BY ADDRESSEE

**Ashton-Tate Periodicals**
PO Box 3729
Escondido, CA 92025-0929

TechNotes
P.O. Box 3729
Escondido, CA 92025-0929

Address Correction Requested

BULK RATE
U.S. POSTAGE
PAID
PERMIT NO. 324
GARDENA, CA